# How to Write Amstrad CPC 464 Games Programs

W. SIMISTER

# HOW TO WRITE
# AMSTRAD CPC464
# GAMES PROGRAMS

## Also by the same Author:

BP157: How to Write ZX Spectrum Games Programs
BP168: How to Write Practical Programs for the Amstrad CPC464

## Other Titles of Interest:

BP152: An Introduction to Z80 Machine Code
BP153: An Introduction to Programming the Amstrad CPC464

# HOW TO WRITE
# AMSTRAD CPC464
# GAMES PROGRAMS

by

W. SIMISTER

BERNARD BABANI (publishing) LTD
THE GRAMPIANS
SHEPHERDS BUSH ROAD
LONDON W6 7NF
ENGLAND

## PLEASE NOTE

Although every care has been taken with the production of this book to ensure that any projects, designs, modifications and/or programs etc. contained herein, operate in a correct and safe manner and also that any components specified are normally available in Great Britain, the Publishers do not accept responsibility in any way for the failure, including fault in design, of any project, design, modification or program to work correctly or to cause damage to any other equipment that it may be connected to or used in conjunction with, or in respect of any other damage or injury that may be so caused, nor do the Publishers accept responsibility in any way for the failure to obtain specified components.

Notice is also given that if equipment that is still under warranty is modified in any way or used or connected with home-built equipment then that warranty may be void.

All the programs in this book have been written and tested by the author using a model of the Amstrad CPC464 that was available at the time of writing in Great Britain. Details of the graphics modes may vary with versions of the machine for other countries.

©1985 BERNARD BABANI (publishing) LTD

First Published – February 1985

British Library Cataloguing in Publication Data
Simister, W.
How to write Amstrad CPC464 games
programs.
1. Computer games. 2. Amstrad CPC464
(Computer) – Programming
1. Title
794.8'028'542 GV1469.2

ISBN 0 85934 134 8

Printed and bounded in Great Britain by Cox & Wyman Ltd, Reading

# FOREWORD

It is expected that readers of this book will have studied the Amstrad 'USER INSTRUCTIONS' book thoroughly, and are now wishing for a step by step guide to writing their own programs. This book will supply that guidance.

Work your way through program after program in the order given in this book, and within a fairly short time a better understanding of the BASIC language used will result. The principle recommended is that the book be studied while sitting at the computer, ready to SAVE all you enter in the so-willing Amstrad Datacorder. In this way the instructions will be absorbed fairly easily.

## PLEASE NOTE

The picture shown on the screen on the cover illustration is simulated.

# CONTENTS

Page

Chapter One
MAKING AND PLACING GRAPHICS CHARACTERS . . . . . . . . . . . . . . . . . . . . . . . . . 1

Chapter Two
WRITING A GAME OF PATIENCE WITH DOMINOES . . . . . . . . . . . . . . . . . . . . . . . . 13

Chapter Three
HOW TO THROW DICE . . . . . . . . . . . . . . . . . . . 26

Chapter Four
A DRAUGHT BOARD AND GAME . . . . . . . . . . . . 38

Chapter Five
REVERSI, USING COMPOSITE U.D.G.s . . . . . . . . . . 51

Chapter Six
THE ANCIENT GAME OF GO . . . . . . . . . . . . . . . . 63

Chapter Seven
A DIFFICULT GAME – CHESS . . . . . . . . . . . . . . . 75

Chapter Eight
PONTOON WITH DOMINOES . . . . . . . . . . . . . . . . 93

Chapter Nine
ORIGIN & DRAW USAGE WITH COLOURS . . . . . . 108

Chapter Ten
A THREE-DIMENSIONAL GAME . . . . . . . . . . . . . 118

Index . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 129

## Chapter One

# MAKING AND PLACING GRAPHIC CHARACTERS

One of the most satisfying moves when starting to master program writing on the Amstrad 464 is to produce a graphic character of your own devising that can be placed anywhere on the screen. Fortunately this is one of the simplest things to do, for it has only three requirements: (i) the ability to reduce a shape to a series of numbers; (ii) a knowledge of how a FOR/NEXT loop operates; and (iii) a knowledge of how to use the LOCATE/PRINT statements.

We will deal with (i) first, but in doing so the other requirements will also be demonstrated (but explained later). Enter this listing: it will produce an illustration on the screen for reference. It will also produce a white screen and black text, which may be easier to understand.

REM stands for REMark, and is ignored by the computer, so it is useful for reference by the programmer.

```
5 REM **GRAPHIC MAKING**
10 REM ****************
15 BORDER 21:INK 1,0:INK 0,26:CLS
20 FOR X=144 TO 400 STEP 32
30 ORIGIN X,96
40 DRAW 0,256
50 NEXT X
60 FOR Y=96 TO 352 STEP 32
70 ORIGIN 144,Y
80 DRAW 256,0
90 NEXT Y
```

Notice, in line 30, the use of ORIGIN instead of PLOT. This ensures that each DRAW statement inside the FOR/NEXT loop has a newly plotted co-ordinate. When this list is typed in key RUN, and then ENTER. A pattern of squares will be produced. There are eight squares by eight squares, and it is in this pattern that all graphic characters are formed. Now add these lines:

```
100 PRINT CHR$(22)+CHR$(1)
110 LOCATE 10,2:PRINT "1":LOCATE 10,
3:PRINT "28643216 8 4 2 1"
115 PRINT CHR$(22)+CHR$(0)
```

It is strongly recommended that the two lines, 100 and 115, (those that include CHR$(22)) are left out until line 110 has been entered and checked. The reason is that line 100 puts the computer into 'Transparent mode', and in this mode EDITing is sometimes difficult. Therefore enter and check line 110, and then enter lines 100 and 115.

It is really essential to take particular notice of the punctuation marks between letters or numbers in all lines when entering them. Comma ( , ), semi-colon ( ; ), colon ( : ), and the inverted commas (" ") must all be meticulously typed in as they are shown in the lists. This applies to spaces, also. For instance, in line 110 there are almost as many punctuation marks and spaces as there are letters and numbers, and each one has its purpose.

In many cases you will only notice the mistake when the computer produces an error message on being told to RUN. Then it is time to check every punctuation mark (as well as letters and numbers) to ensure that they are all correct.

Lines 100 to 115 produce some numbers at the top of the pattern of squares. They represent the values of the columns of squares below them. The first one (from the left) is 128, the second 64, then 32, 16, 8, 4, 2, 1. These are more easily remembered if you start from the right: 1; double that 2; double that 4; double that 8; and so on to 128, which is double 64. Now enter this larger block of lines.

```
120 LOCATE 16,4:PRINT CHR$(143);CHR$
(143):LOCATE 16,5:PRINT CHR$(143);CH
R$(143)
130 LOCATE 14,6:PRINT CHR$(143);CHR$
```

```
(143):LOCATE 14,7:PRINT CHR$(143);CH
R$(143)
140 LOCATE 12,8:PRINT CHR$(143);CHR$
(143):LOCATE 12,9:PRINT CHR$(143);CH
R$(143)
150 LOCATE 10,10:PRINT CHR$(143);CHR
$(143);CHR$(143);CHR$(143);CHR$(143)
;CHR$(143)
160 LOCATE 10,11:PRINT CHR$(143);CHR
$(143);CHR$(143);CHR$(143);CHR$(143)
;CHR$(143)
170 LOCATE 12,12:PRINT CHR$(143);CHR
$(143);CHR$(143);CHR$(143);CHR$(143)
;CHR$(143);CHR$(143);CHR$(143);CHR$(
143);CHR$(143);CHR$(143);CHR$(143);C
HR$(143);CHR$(143)
180 LOCATE 12,13:PRINT CHR$(143);CHR
$(143);CHR$(143);CHR$(143);CHR$(143)
;CHR$(143);CHR$(143);CHR$(143);CHR$(
143);CHR$(143);CHR$(143);CHR$(143);C
HR$(143);CHR$(143)
190 LOCATE 14,14:PRINT CHR$(143);CHR
$(143);CHR$(143);CHR$(143);CHR$(143)
;CHR$(143);CHR$(143);CHR$(143);CHR$(
143);CHR$(143);CHR$(143);CHR$(143)
200 LOCATE 14,15:PRINT CHR$(143);CHR
$(143);CHR$(143);CHR$(143);CHR$(143)
;CHR$(143);CHR$(143);CHR$(143);CHR$(
143);CHR$(143);CHR$(143);CHR$(143)
210 LOCATE 14,16:PRINT CHR$(143);CHR
$(143);CHR$(143);CHR$(143):LOCATE 22
,16:PRINT CHR$(143);CHR$(143)
220 LOCATE 14,17:PRINT CHR$(143);CHR
$(143);CHR$(143);CHR$(143):LOCATE 22
,17:PRINT CHR$(143);CHR$(143)
230 LOCATE 12,18:PRINT CHR$(143);CHR
```

```
$(143):LOCATE 16,18:PRINT CHR$(143);
CHR$(143):LOCATE 22,18:PRINT CHR$(14
3);CHR$(143)
240 LOCATE 12,19:PRINT CHR$(143);CHR
$(143):LOCATE 16,19:PRINT CHR$(143);
CHR$(143):LOCATE 22,19:PRINT CHR$(14
3);CHR$(143)
```

CHR$(143) is an existing solid block character, and it is used in these lines to fill some squares in the pattern. Four characters are needed to fill each square in the form we have used to draw it (to make it large on the screen), so there are a lot to enter. Make sure of all the punctuation marks, remembering that a colon ( : ) separates off two statements or commands, and a semi-colon ( ; ) tells the computer to position the next character closely to the previous one. Mistakes in these will cause an error message to appear when they are RUN.

If entered correctly the pattern that is produced on the screen is that of a deer, quite crude in this enlarged version but credible when small. Enter the next four lines using the same method as before for lines containing CHR$(22) etc. That means: enter lines 250 and 260, check that they are correct, and then enter lines 245 to 265.

```
245 PRINT CHR$(22)+CHR$(1)
250 LOCATE 27,5:PRINT "16":LOCATE 27
,7:PRINT "32":LOCATE 27,9:PRINT "64"
:LOCATE 26,11:PRINT "224"
260 LOCATE 26,13:PRINT "127":LOCATE
27,15:PRINT "63":LOCATE 27,17:PRINT
"50":LOCATE 27,19:PRINT "82"
265 PRINT CHR$(22)+CHR$(0)
```

When the program is now RUN there will be shown the pattern of the deer, the numbers showing the column values at the top, and the row values at the right. These row values are obtained by totalling the values of each filled in square in that row.

For instance: row one has a black square under 16. Enter that on the right. Row two has a black square under 32. Enter that on the right. Row three has one under 64. Enter that. Row four has three black squares under 128, 64, and 32. Adding these together gives 224. Enter that on the right. The same procedure continues to the eighth row, and if one happens to be a blank row, then that is entered as 0.

To emphasize this point use a blank sheet of paper, and draw on it a pattern of 8 x 8 squares (like the one on the screen), and fill in some of the squares to make a pattern. Here is an example you might like to follow: put three black squares in the first row under the numbers 128, 64, and 32. These total 224, so enter that at the right for row 1. In the second row enter one square under 16. Put that on the right. In the third row enter one square under 32, and enter that. In the fourth row enter five black squares under 64, 32, 16, 8, and 4. These add together to make 124; enter that on the right.

Row five has six squares under 64, 32, 16, 8, 4, and 2, so enter 126 for that. Row six is under 32, 16, 8, 4, 2, and 1, making 63 to enter on the right. Row seven is under 8, 4, 2, and 1, so enter 15. The last row (8) is just one square under 4; so enter that.

This pattern makes a goose, and the eight numbers are used to form a graphic character, just as the first example made a deer. Enter these nine lines now.

```
270 SYMBOL AFTER 240
280 SYMBOL 241,16,32,64,224,127,63,5
0,82:REM DEER
290 LOCATE 32,9:PRINT CHR$(241)
300 SYMBOL 242,224,16,32,124,126,63,
15,4:REM GOOSE
310 LOCATE 32,11:PRINT CHR$(242)
320 REM
330 REM
340 REM
350 REM
```

The four empty REM rows are there for you to fill with experiments of your own. Graphic characters on the Amstrad 464 are extremely easy to produce. In the User Instruction book at Appendix 3, pages 2 to 13, can be found a set of over 200 graphic characters ready to use with: PRINT CHR$(33) etc. Each of these characters can be re-defined to one of your own design using a special formula.

First of all decide how many characters you are going to require. Then, avoiding those already existing which you may want to use (we have already used CHR$(143)) you select a block to use for yourself: the block must be a continuous one. I have chosen CHR$(241) for the first one in this program.

The formula is: (line No.) SYMBOL AFTER 241 (this we chose). (next line) SYMBOL 241, (and then eight numbers like those we explained above). Each one must be separated by a comma. Lines 270, 280, and 300 show this in practice. Lines 290 and 310 show how to write a line to print them on the screen. To repeat the formula: first line: SYMBOL AFTER (a number); second line: SYMBOL (number), (and the eight numbers divided by commas). Put a REM (separated from the last number by a colon) at the end of each SYMBOL line for your own reference later.

When the program is now RUN the two characters will appear to the right of the diagram. These two characters (and any others you may design) will be available in the computer until it is turned off. Even using NEW will not remove them.

Having carefully studied the results obtained so far you will realise that with a sheet of squared paper almost any kind of character can be made. Remember, however, always use a single block of numbers starting with SYMBOL AFTER (whichever number you decide to use).

It is now time to examine the LOCATE/PRINT statements. For this we shall use another screen, taking the opportunity to change the border colour, also.

Notice that in line 360 the INPUT position has been located near the bottom; left alone it would appear higher up. The 'variable' given to the reply is J$, so that line 370 is easy to understand. Line 390 is there merely to prevent any move without clearing the screen. CLS in lines 370 and 520 will do that.

```
360 LOCATE 2,24:INPUT "ENTER 2 FOR N
EXT SECTION.";J$
370 IF J$="2" THEN CLS:GOTO 500
390 LOCATE 2,22:STOP
500 REM **LOCATE**
510 REM **********
520 BORDER 17:CLS
```

At this stage SAVE all you have entered. I make a practice of doing this after every few lines (well, a dozen or so lines) because in my area we are liable to power cuts at odd times. Our supply comes from automatically controlled transformers. Yours may be the same without you being aware of it, so keep on recording as you work, it is easy with the Amstrad. Now enter these lines.

```
530 FOR X=1 TO 640 STEP 16
540 ORIGIN X,1
550 DRAW 1,400
560 NEXT X
570 FOR Y=1 TO 400 STEP 16
580 ORIGIN 1,Y
590 DRAW 640,1
600 NEXT Y
```

To start off this section we use FOR/NEXT loops that enable the Amstrad to print parallel lines, using ORIGIN instead of PLOT. To see the result of PLOT instead of ORIGIN, EDIT line 540, and change ORIGIN to PLOT. When this is RUN (RUN 500) the result is that the vertical lines become a series of slanting ones near the top of the screen. They would converge at a single point if they were lower on the screen. Now EDIT 540 once more, and replace origin. This little experiment will show that when using a FOR/NEXT loop to repeat parallel lines, ORIGIN must be used.

With that pattern of squares now in the program we must enter four more lines. These include the Transparent mode again CHR$(22)+(CHR$(1)) so first enter lines 620 and

630. RUN 500 again to see that they are correct, and then enter lines 610 and 640 to make them print without erasing any background.

```
610 PRINT CHR$(22)+CHR$(1)
620 LOCATE 1,1:PRINT "123456789-&-ON
-TO-40"
630 LOCATE 1,2:PRINT "2":PRINT "3":P
RINT "4":PRINT "5":PRINT "6":PRINT "
7":PRINT "8":PRINT "9":PRINT "-":PRI
NT "&":PRINT "-":PRINT "O":PRINT "N"
:PRINT "-":PRINT "T":PRINT "O":PRINT
 "-":PRINT "2":PRINT "5"
640 PRINT CHR$(22)+CHR$(0)
```

With these lines the X numbers at the top, and the Y numbers down the left hand side (used in LOCATE) are put into their places. Beyond 9 the numbers would occupy two spaces each, so they are not included. It will be realised by now (if you entered and tested lines 620 and 630 on their own first) just how effective that Transparent command really is.

Another point to be remembered is the use of inverted commas (“ ”) to enclose whatever is to be printed. Without those signs the computer would interpret (literally) what it was told to print, and, for instance, dash signs ( – ) would become minus signs. The computer prints (as it stands) whatever is enclosed by inverted commas. In line 630 the colons ( : ) between each PRINT statement ensure that each succeeding command is printed one line lower.

```
650 LOCATE 6,5:PRINT CHR$(241):LOCAT
E 16,15:PRINT CHR$(242)
660 LOCATE 7,7:PRINT "THIS IS A TEST
"
670 PRINT CHR$(22)+CHR$(1)
680 LOCATE 7,9:PRINT "THIS IS A TEST
"
690 PRINT CHR$(22)+CHR$(0)
```

This small section prints the deer and the goose at different places, without the Transparent mode. Leave out lines 670 and 690 until the others are entered and checked with RUN 500. Then add those two lines, and note the difference once more. With this demonstration you should find it easy to add lines of your own after line 690. Do this as a practice, and then we must move on to the next section.

To examine the FOR/NEXT loop more closely we will need another screen, starting at 1000. Save all you have done, and then add these lines.

```
800 LOCATE 2,24:INPUT "ENTER 3 FOR N
EXT SECTION.";K$
810 IF K$="3" THEN CLS:GOTO 1000
990 STOP
1000 REM *FOR-NEXT  LOOPS*
1010 REM * * * * * * * * *
1020 BORDER 15:CLS
1030 X=1:FOR Y=1 TO 25
1040 LOCATE X,Y:PRINT CHR$(241)
1050 NEXT Y
```

This produces a different border colour, and a column of deer at the left hand side when RUN 1000 is entered. Enter three more lines now. They are similar to those at 1030–1050, except for two slight differences. See if you can spot them.

```
1060 X=3:FOR Y=1 TO 25 STEP 2
1070 LOCATE X,Y:PRINT CHR$(241)
1080 NEXT Y
```

X has now become 3 instead of 1, and ‘STEP 2’ has been added to the ‘FOR Y’ part of the statement. It is that STEP 2 that has spaced them out. Add another four lines, and examine the differences in these.

```
1090 FOR X=5 TO 40 STEP 2
1100 FOR Y=1 TO 25 STEP 2
1110 LOCATE X,Y:PRINT CHR$(242)
1120 NEXT Y:NEXT X
```

Before going any farther SAVE all the program onto tape. When this has been done RUN 1000. There will be a screenful of deer and geese. The explanation of how a FOR/NEXT loop produces this result needs careful study.

There are four FOR/NEXT loops between lines 1030 and 1120, the last two being between 1090 and 1120, where the Y loop is enclosed by the X loop. This is called 'NESTING', but more of that later. Starting with the statement at line 1030 'X=1: FOR Y=1 TO 25', and finishing at line 1050 'NEXT Y', the computer reads X=1, so knows that whatever it is commanded to do must be done in column 1. It reads 'FOR Y=1 TO 25' and knows that it has to repeat the PRINT at all rows from 1 to 25. It keeps repeating the PRINT at X(1), Y(1 TO 25) so long as a Y number remains, and when Y is finished it goes to the next line 1060.

There, in line 1060, it meets a different formula, including 'STEP 2'. This means that it must repeat the PRINTing on alternate rows only, at X column 3. When Y runs out of numbers the computer goes on to line 1090, where there are two NESTED loops.

The word 'NESTED' means that one loop must be completely enclosed by the other. At column 5(X) the computer PRINTS the character (a goose this time) at the places specified by the Y loop (1 TO 25 STEP 2). This is at each place in column 5 that Y specifies. Having done this it goes back to the next X column (7) and fills that in. And so it continues until the Y numbers are exhausted, and so are those of X, and it can pass on to the next line number. It does all this very quickly, but if you watch carefully you will be able to see it in operation.

FOR/NEXT loops can be used for a number of repetitive purposes, and making a series of sounds is one example. The sounds produced by the Amstrad are fairly complex, but here we won't deal with them in detail. A range of simple tones will suffice. Enter these three lines now.

```
1130 FOR X=100 TO 50 STEP -5
1140 SOUND 1,X,10
1150 NEXT X
```

In the FOR X line the pitch of the tone is made to rise in steps of 5 from 100 to 50 (lower numbers are higher pitched) and repeats the tone at a higher pitch as it goes on. When it reaches 50 it finishes. Note the way in which this entry is made: SOUND 1,X,10. It will be realised that the central number controls the note. This sound can be made to repeat with the aid of another FOR/NEXT loop nested around the first one. Enter these extra lines and modify line 1140 as shown here.

```
1125 FOR Z=1 TO 3
1130 FOR X=100 TO 50 STEP -5
1140 SOUND 1,X+Z,10
1150 NEXT X
1155 NEXT Z
```

The Z loop is now nested around the X loop, and so the tune is repeated three times. This is a simple illustration of how to use a nested loop. Note that the X loop governs the tune, and the Z loop governs the number of times it is played.

If you have followed the instructions so far you have now written a program consisting of three separate demonstrations of what you will need to know to tackle the games programs that follow. Now we will add a few lines so that we will be able to switch back to screen 1 again.

```
1400 LOCATE 1,25:INPUT "ENTER 1 FOR
FIRST SECTION.";L$
1410 IF L$="1" THEN CLS:GOTO 10
```

With the aid of this small arrangement you will be able to switch to any of the screens in this program at will. To become free from this sequences, use ESC. Then you can use LIST, RUN, or SAVE as you wish.

Incidentally, when wishing to LIST lines in the middle of a program, say at 500, then enter LIST 500–. Don't forget that dash, for without it only the one line would be shown on the screen. With it the whole program after that line number is available. That minus sign means 'onward' in this context, so be ready with the ESC button to stop it where you want it.

If you have just skipped through this chapter, wanting to get to the games programs, then be warned: without the knowledge in this first chapter you will find the game heavy going.

# Chapter Two

## WRITING A GAME OF PATIENCE WITH DOMINOES

If you have thoroughly studied and worked through the first chapter there is now nothing between your desire to write a complete games program for the Amstrad, and its completion except a deal of thinking, and a patient following of the instructions in this chapter.

I have chosen a game of Domino Patience for this example, mainly because it entails the use of many things you have learnt so far, and also because with a computer you can use many more dominoes than are available with a normal set.

There will have to be many sections in such a program, for one will be needed to produce the dominoes, and another to play the game. It would also be advisable to have a screenful of rules before the game starts, so there must be a section for that, and another to lay out the screen for the actual game. We shall start by producing the dominoes.

When a computer has been instructed to produce some dominoes in graphic characters, they are stored in memory, and will always be available during the course of successive games, so there is no need for this section to be near the beginning as it takes a little time to run through its procedure, and this would introduce an unnecessary delay if run through regularly.

Therefore we use a routine called GOSUB/RETURN, which will allow us to place the graphic making section at the end of the program. A skeleton will be written first, into which the flesh of the program can be fitted. A section will be used for rules, another for screen lay-out, and another for the actual moves. Enter these lines now.

```
10 REM DOMINO PATIENCE
20 REM ***************
30 BORDER 7:INK 1,0:INK 0,26:CLS
40 GOSUB 5000
50 GOSUB 4000
```

```
60 GOSUB 3000
1000 REM GAME
1010 REM ****
2999 STOP
3000 REM GAME SCREEN
3010 REM ***********
3998 RETURN
3999 STOP
4000 REM RULES SCREEN
4010 REM ************
4998 RETURN
4999 STOP
5000 REM MAKE DOMINOES
5010 REM *************
5998 RETURN
5999 STOP
```

When this list is studied it will be seen to form a framework for the program. Some parts need explaining, however. Line 40 GOSUB 5000 sends the computer to line 5000, and 5998 sends it back to the line **immediately after** the one that has sent it to 5000, as the computer always remembers correctly where it came from GOSUB/RETURN can always be relied on.

Therefore line 50 sends it to 4000 (the Rules), 4998 sends it back to the line after 50 and so on. The STOP lines at 2999, 3999, 4999, and 5999 are inserted only for our convenience. They prevent the computer entering a following section if we should lose a RETURN from anywhere. Having now got a good skeleton we shall work on each section in turn, starting at 5000.

We shall use graphic characters to form the dominoes, so that there is a black domino with white pips. However, because a domino is oblong, while graphic characters are square, we shall need two graphic for each domino. Fortunately we can use the same seven characters. Enter these lines now.

```
5020 SYMBOL AFTER 240
5030 SYMBOL 240,255,255,255,255,255,
255,255,0:REM DOM 0
5040 SYMBOL 241,255,255,255,231,255,
255,255,0:REM DOM 1
5050 SYMBOL 242,255,159,255,255,255,
249,255,0:REM DOM 2
5060 SYMBOL 243,255,159,255,231,255,
249,255,0:REM DOM 3
5070 SYMBOL 244,255,153,255,255,255,
153,255,0:REM DOM 4
5080 SYMBOL 245,255,153,255,231,255,
153,255,0:REM DOM 5
5090 SYMBOL 246,255,153,255,153,255,
153,255,0:REM DOM 6
```

When these lines are entered it may be noticed that the last number of the SYMBOL numbers (240 to 246) appear to coincide with the domino pip values. This is no coincidence, for I started the numbering at 240 with this in mind. It will make the task easier later on. The formula used in lines 5030 to 5090 (making the half dominoes) is perfectly straightforward. The line number is followed by ‘SYMBOL’ and then the character number. Then, after a comma, and spaced with commas, come the eight numbers forming that character (nine numbers in all, each separated from the other by a comma). The REM at the end of each line is for your own reference, it is ignored by the computer.

To check that the half dominoes have been correctly entered insert this temporary line: 1020 LOCATE 3,10: PRINT CHR$(240);“ ”;CHR$(241);“ ”;CHR$(242);“ ”; CHR$(243);“ ”;CHR$(244);“ ”;CHR$(245);“ ”;CHR$(246). Now RUN the program, and this should appear on your screen as seven half dominoes. If correct, then enter 1020 on its own to delete it.

The graphics section is now complete, so we shall proceed with the RULES section at 4000. Because the game of Domino Patience is not well known (it is my own invention) it is necessary to produce a display of what the completed game should look like. The dominoes must be located in the positions that they will occupy when playing the game, so to do this you must study the LOCATE statement once more.

Along the top of the screen are the X positions 1 to 40, and down the left hand side are the Y positions 1 to 25. When positions on the screen have to be labelled for play (as in this game) it is useful to use letters instead of numbers (mainly because numbers larger than 9 occupy two spaces). In this particular case each position is three spaces from its neighbour, and irregularly spaced numbers could be confusing (4, 7, 10, 13, etc.). Therefore letters are used, and converted to an X or Y value during the program. Enter these lines now.

```
4020 LOCATE 4,2:PRINT "A  B  C  D  E
  F  G"
4030 LOCATE 2,4:PRINT "A":LOCATE 2,7
:PRINT "B":LOCATE 2,10:PRINT "C":LOC
ATE 2,13:PRINT "D":LOCATE 2,16:PRINT
 "E":LOCATE 2,19:PRINT "F":LOCATE 2,
22:PRINT "G"
4040 A=4:B=7:C=10:D=13:E=16:F=19:G=2
2
```

The domino positions have been located so that A, in either X or Y position, will equal 4. Lines 4020 and 4030 position the A,B,C numbers, and line 4040 gives them a value. The seven letters (A to G) are now variables, and must not be used as variables for anything else in this program. RUN the program now so that those values are put into the memory. The computer will understand from now on that LOCATE D,C: means to LOCATE 13,10: Try entering (without any line number): LOCATE C,B:PRINT CHR$(244);CHR$(243) and then ENTER. The domino 4–3 will appear at that location. This will not harm the program, and will disappear without trace at the next LIST or RUN.

It is now necessary to tell the computer which domino to place in which position for the display, so enter this:

```
4050 LOCATE A,A:PRINT CHR$(246);CHR$
(246):LOCATE A,B:PRINT CHR$(246);CHR
$(245):LOCATE A,C:PRINT CHR$(246);CH
R$(244):LOCATE A,D:PRINT CHR$(246);C
HR$(243)
4060 LOCATE A,E:PRINT CHR$(246);CHR$
(242):LOCATE A,F:PRINT CHR$(246);CHR
$(241):LOCATE A,G:PRINT CHR$(246);CH
R$(240)
4070 LOCATE B,A:PRINT CHR$(245);CHR$
(245):LOCATE B,B:PRINT CHR$(245);CHR
$(244):LOCATE B,C:PRINT CHR$(245);CH
R$(243)
4080 LOCATE B,D:PRINT CHR$(245);CHR$
(242):LOCATE B,E:PRINT CHR$(245);CHR
$(241):LOCATE B,F:PRINT CHR$(245);CH
R$(240)
4090 LOCATE C,A:PRINT CHR$(244);CHR$
(244):LOCATE C,B:PRINT CHR$(244);CHR
$(243):LOCATE C,C:PRINT CHR$(244);CH
R$(242)
4100 LOCATE C,D:PRINT CHR$(244);CHR$
(241):LOCATE C,E:PRINT CHR$(244);CHR
$(240)
4110 LOCATE D,A:PRINT CHR$(243);CHR$
(243):LOCATE D,B:PRINT CHR$(243);CHR
$(242):LOCATE D,C:PRINT CHR$(243);CH
R$(241):LOCATE D,D:PRINT CHR$(243)CH
R$(240)
4120 LOCATE E,A:PRINT CHR$(242);CHR$
(242):LOCATE E,B:PRINT CHR$(242);CHR
$(241):LOCATE E,C:PRINT CHR$(242);CH
R$(240)
4130 LOCATE F,A:PRINT CHR$(241);CHR$
(241):LOCATE F,B:PRINT CHR$(241);CHR
$(240):LOCATE G,A:PRINT CHR$(240);CH
R$(240)
```

This is a long, tedious list to enter, but the advantage of having the last number of the SYMBOL number the same as the domino it forms will now be apparent: it is easier to remember the number that goes between the brackets after CHR$. When this is correctly entered it will print up on the screen, and we can proceed to enter the game title, the instructions, and a X to mark where the domino will appear when the game starts.

Notice, in line 4150, that PEN 3: is used to print the title in red. If this was left unaltered everything else from then on would be red; therefore, in line 4160, PEN 1: is used to reverse this. PEN 3: and PEN 1: are used again for the INPUT line (lines 4200 and 4210). Line 4190 reminds the player to go into CAPS mode.

```
4140 LOCATE 11,22:PRINT "X"
4150 PEN 3:LOCATE 28,2:PRINT "*DOMIN
O*":LOCATE 28,4:PRINT "PATIENCE"
4160 PEN 1:LOCATE 25,6:PRINT "AIM AT
 THIS":LOCATE 25,7:PRINT "PATTERN."
4170 LOCATE 23,9:PRINT "IF DOMINO AT
 X IS":LOCATE 23,10:PRINT "A DOUBLE
THEN USE":LOCATE 23,11:PRINT "IT TO
START."
4180 LOCATE 21,13:PRINT "ONLY ENTER
DOMINOES":LOCATE 21,14:PRINT "IN SEQ
UENCE AS":LOCATE 21,15:PRINT "SHOWN
HERE."
4190 LOCATE 19,18:PRINT "PRESS CAPS
LOCK."
4200 PEN 3:LOCATE 19,24:INPUT "ENTER
 P TO START";Q$
4210 PEN 1:IF Q$="P" THEN CLS:RETURN
4220 IF Q$<>"P" THEN GOTO 4200
```

The INPUT lines 4200–4220 change the screen to that at 3000, the GAME screen; although if you RUN the program now you will receive the signal ‘Break in 2999’. This is because there is nothing at 3000 to stop the RETURN at

3998, and so it continues to the GAME section at 1000, where line 2999 comes into operation.

We will now start to plan the GAME screen. A title will be needed; a score section; and some more instructions; as well as the position letters, and some boxes in which to LOCATE each of the dominoes. That must be done first.

Beginners at programming will be wondering by now how it is possible to LOCATE text on a screen at suitable positions, and may think it is done by trial and error. This is not so. It is done with the aid of a plotting board. Such things are advertised in computer magazines, but you can easily make one yourself. My first one was made on a piece of stiff cardboard, and it lasted for some time.

On the board are ruled 26 horizontal and 41 vertical lines spaced about ¼ inch apart. These produce 1000 squares (40 along the top, and 25 down the side). That is the number of spaces on the Amstrad 464 in Mode 1 (the mode in which I prefer to work). I extend all the lines a little beyond the boundary line, so that the numbering of the positions is clearer. A short program will illustrate this better than many words. It will not alter the program at all, being located at lines 8000–8140, and it will be another exercise in programming.

Incidentally, if you have not realised it yet, it is possible to summon up any portion of a program list by entering LIST and the line number followed by a dash: 'LIST 8000–' in this case. When you do this in a longer section of program have a finger near the ESC button in order to stop the scrolling where you want it.

```
8000 REM TEST
8010 FOR X=184 TO 494 STEP 32
8020 ORIGIN X,52
8030 DRAW 0,320
8040 NEXT X
8080 FOR Y=68 TO 372 STEP 32
8090 ORIGIN 164,Y
8100 DRAW 324,0
8110 NEXT Y
```

```
8120 LOCATE 13,2:PRINT "1 2 3 4 5 6
7 8 9":LOCATE 20,1:PRINT "X"
8130 LOCATE 11,4:PRINT "1":LOCATE 11
,6:PRINT "2":LOCATE 11,8:PRINT "3":L
OCATE 11,10:PRINT "4":LOCATE 8,12:PR
INT "Y  5"
8140 LOCATE 11,14:PRINT "6":LOCATE 1
1,16:PRINT "7":LOCATE 11,18:PRINT "8
":LOCATE 11,20:PRINT "9"
```

When this is entered, and you RUN 8000, it will show the top left hand corner of the board you are designing. in this enlarged version it is only possible to give 9 numbers each way, but it does illustrate what has been described so far.

The plotting board you are making will have 40 horizontal spaces and 25 vertical ones, so the X numbers will go (left to right) 1 to 40, while the Y numbers will go (top to bottom) 1 to 25.

In addition to these, for the PLOT/ORIGIN/DRAW commands you will require the PIXEL numbers: X along the bottom from left to right, and the Y numbers up the right hand side. These numbers progress in steps of 16: 0, 16, 32, 48, 64, 80, and so on to 640 at the bottom, and to 400 up the right hand side. These numbers are at the line ends, not the spaces in between, starting with 0 for the border line in each case.

When the board is complete in ink, of course any light pencil mark can be removed easily with a rubber; so it is easy to enter in all that has to go onto the screen, remembering that each space represents a letter, or a space, or a comma, etc. I find it nearly impossible to do any serious programming without such a board, and now make mine on a sheet of Formica as it lasts longer, and as the ink lines become faded they are easily re-drawn. Try using Indian ink for permanence.

Before this plotting board diversion we were about to draw the lines forming the boxes for the domino game. These, being all of different sizes, must be entered separately; hence the long list of ORIGIN/DRAW statements in lines 3020–3050. However, the lines used in section 4000 for the location letters and variables can soon be transferred to this section by

EDITing their line numbers, so that will ease the problem a little.

```
3020 ORIGIN 40,32:DRAW 0,335:ORIGIN
88,32:DRAW 0,335:ORIGIN 136,80:DRAW
0,287:ORIGIN 184,128:DRAW 0,239
3030 ORIGIN 232,176:DRAW 0,191:ORIGI
N 280,224:DRAW 0,143:ORIGIN 328,272:
DRAW 0,96:ORIGIN 376,320:DRAW 0,47
3040 ORIGIN 40,367:DRAW 336,0:ORIGIN
 40,320:DRAW 336,0:ORIGIN 40,272:DRA
W 288,0:ORIGIN 40,224:DRAW 240,0
3050 ORIGIN 40,176:DRAW 192,0:ORIGIN
 40,128:DRAW 144,0:ORIGIN 40,80:DRAW
 96,0:ORIGIN 40,32:DRAW 48,0
3060 LOCATE 4,2:PRINT "A  B  C  D  E
  F  G"
3070 LOCATE 2,4:PRINT "A":LOCATE 2,7
:PRINT "B":LOCATE 2,10:PRINT "C":LOC
ATE 2,13:PRINT "D":LOCATE 2,16:PRINT
 "E":LOCATE 2,19:PRINT "F":LOCATE 2,
22:PRINT "G"
3080 A=4:B=7:C=10:D=13:E=16:F=19:G=2
2
```

Entering the rest of the Game screen section is simple, provided you have a plotting board on which to locate the text. Having done this myself for this game, I have now only to read off the locations, and enter them for printing. One other point, though, is the introduction of some colour, and I have done this with three lines, as will be seen in the following list.

```
3090 LOCATE 11,22:PRINT "X"
3100 PEN 3:LOCATE 30,2:PRINT "SCORE"
:PEN 1
3110 LOCATE 31,4:PRINT "FOR:":LOCATE
 27,6:PRINT "AGAINST:":LOCATE 27,8:P
RINT "AVERAGE:"
```

```
3120 PEN 3:LOCATE 26,10:PRINT "*****
*****":LOCATE 26,11:PRINT "**DOMINO*
*":LOCATE 26,12:PRINT "*PATIENCE*":L
OCATE 26,13:PRINT "**********":PEN 1
3130 LOCATE 19,15:PRINT "FIRST ENTER
 Y OR N"
3140 LOCATE 17,17:PRINT "THEN TOP LE
TTER":LOCATE 17,18:PRINT "AND SIDE L
ETTER"
3150 PEN 3:LOCATE 17,19:PRINT "***SE
PARATELY**":PEN 1
```

Line 3100 shows PEN 3 (red) at the line beginning, and PEN 1 at its end. This applies also to lines 3120, and 3150. Isolating the colour to its own line in this fashion seems to be the most satisfactory way. The games screen is now complete, and the game itself, at 1000, can be planned.

We start this by making some two-letter 'variables' for the SCORE, and printing an 0 at these positions. It is followed by a restriction on the number of dominoes to be used.

```
1020 FX=0:AX=0:VE=0:LOCATE 35,4:PRIN
T FX:LOCATE 35,6:PRINT AX:LOCATE 35,
8:PRINT VE
1030 IF AX=225 THEN LOCATE 18,22:PRI
NT "GAME ENDED.":FOR X=100 TO 50 STE
P-5:SOUND 1,X,20:NEXT X: GOTO 1150
```

The next few moves are crucial to the whole game, for we have to RANDOMIZE (produce different varieties of) the domino halves, and see that they change at each move. The RND command is inclined to repeat a series of changes, and can only be called 'Pseudo-random'. Using RANDOMIZE in front of them improves the situation, and RANDOMIZE TIME is even better. We will use that. The domino halves must be put into a string, so that the two halves can be joined and then printed together.

```
1040 A$=CHR$(240)+CHR$(241)+CHR$(242
)+CHR$(243)+CHR$(244)+CHR$(245)+CHR$
(246)
1050 B$=CHR$(240)+CHR$(241)+CHR$(242
)+CHR$(243)+CHR$(244)+CHR$(245)+CHR$
(246)
1060 RANDOMIZE TIME
1070 Z=INT(RND*7)+1
1080 Q=INT(RND*7)+1
1090 LOCATE 13,22:PRINT MID$(A$,Z,1)
+MID$(B$,Q,1)
```

When this has been entered and checked, RUN the program, and, after pressing P to replace the Rules with the Game screen, a domino should be printed at X. Do this a few times to make sure it changes at every RUN. If it does not, then something has been entered wrongly.

```
1100 K$=MID$(A$,Z,1)+MID$(B$,Q,1)
1110 LOCATE 2,25:INPUT "CAN YOU GO(Y
/N)";E$
1120 IF E$="N" THEN AX=AX+1:LOCATE 3
5,6:PRINT AX:GOTO 1030
1130 IF E$="Y" THEN FX=FX+1:LOCATE 3
5,4:PRINT FX:GOTO 1180
1140 IF E$<>"N" OR E$<>"Y" THEN GOTO
 1110
1150 LOCATE 35,8:PRINT "1:";INT(AX/F
X)"
1160 LOCATE 2,25:INPUT "ENTER R FOR
ANOTHER GAME";P$
1170 IF P$="R" THEN CLS:GOTO 60
```

Line 1100 now makes a string (K$) out of the two combined RANDOMIZED domino halves, and the next few lines begin to play the game with an INPUT line (1110) which gives the answer (Y/N) the variables E$. The following lines either go back to 1030 to repeat the domino presentation at X, or to 1180, whilst adding to the score. At 1150 is a routine for use at the end of the game; with 1170 sending the computer back to line 60 to re-draw the game screen from scratch.

```
1180 LOCATE 2,25:PRINT "
               ":LOCATE 2,25:INPUT "
WHERE(TOP)";X$
1190 LOCATE 2,25:PRINT "
               ":LOCATE 2,25:INPUT "
WHERE(SIDE)";Y$
1200 IF X$="A" THEN X=4
1205 IF X$="B" THEN X=7
1210 IF X$="C" THEN X=10
1215 IF X$="D" THEN X=13
1220 IF X$="E" THEN X=16
1225 IF X$="F" THEN X=19
1230 IF X$="G" THEN X=22
1240 IF Y$="A" THEN Y=4
1245 IF Y$="B" THEN Y=7
1250 IF Y$="C" THEN Y=10
1255 IF Y$="D" THEN Y=13
1260 IF Y$="E" THEN Y=16
1265 IF Y$="F" THEN Y=19
1270 IF Y$="G" THEN Y=22
1280 LOCATE X,Y:PRINT K$:GOTO 1290
1290 LOCATE 12,22:PRINT "     ":FOR
X=100 TO 50 STEP -5:SOUND 1,X,10:NEX
T X:GOTO 1060
```

In line 1130 a Y answer sends the computer to 1180. At that line is the beginning of a routine that recognises the domino at X which you want to use in the pattern, and sends it to the location you stipulate when answering the INPUTS at lines 1180 and 1190. Line 1280 actually prints the domino; line 1290 plays a tune and sends the computer back to 1060 to produce another random domino at X.

This looping routine continues until you have used up 225 dominoes (you can alter the number in line 1030 if you wish).

At 225 dominoes the computer goes to the routine at 1150, prints the average score (INT(AX/FX)), and then asks if you want another game. Line 1170 (if R is answered) clears the screen and goes to 60.

With these entries the game is virtually complete, but before you start playing there are a few lines that can now be erased, for their purpose (to prevent the computer going past a section while the program is incomplete) is now finished. Lines 2999, 3999, and 4999 can now be deleted. You may wish to retain the program at 8000 for reference, it won't affect the game itself. If not, then delete these lines as well.

The game should now play correctly, and you will find that with line 1030 set at 'AX=225', you have an equal chance of winning or losing. That is an average of 1 placed out of every 8 dominoes used. Reducing AX to 200, makes it 1 in 7 dominoes – much harder. Of course, by cheating you can win every time; but then, what is the point of that?

One final point: if you do lose, the computer tells you so, and offers you another go. If you win there is no response, and you will have to use ESC and then RUN to start all over again. If you have advanced sufficiently in your programming to do so, why not try to include a routine that makes the computer acknowledge the winner? Try it as a project it should not be too difficult. Remember, however, to put the whole game onto tape before you start altering it. Then, if anything does go wrong, you can clear the computer, and start again with LOAD " " .

## Chapter Three

# HOW TO THROW DICE

While working on the game of Domino Patience it occurred to me that a half domino would form a good Die (the singular of dice). Then I remembered an old game of 'Chuck-a-Luck', in which three dice were thrown. You bet on a number, and if it turned up once you got your money back; if twice, then double your money; and if thrice, then treble your money. It would make a nice computer game, and would divide easily into sections to form a skeleton, such as we used last time.

A graphics section for making the dice can be borrowed from Domino Patience, using only the half domino, and without the blank. That could start at 5000. The screen can be laid out in section 4000. The dice must be thrown to their positions at line 3000. The dice must be converted to ordinary numbers for scoring in section 2000, and the moves of the game could start at line 1000. That produces the following skeleton.

```
10 REM CHUCK-A-LUCK
20 REM ************
30 BORDER 8:INK 1,0:INK 0,24:CLS
40 GOSUB 5000
50 GOSUB 4000
60 GOSUB 3000
70 GOSUB 2000
80 GOSUB 1000
1000 REM GAME
1010 REM ****
1990 LOCATE 2,22:STOP
2000 REM DICE TO NUMBERS
2010 REM ***************
2999 RETURN
3000 REM THROWING DICE
3010 REM *************
```

```
3999 RETURN
4000 REM GAME SCREEN
4010 REM ***********
4999 RETURN
5000 REM GRAPHICS
5010 REM ********
5999 RETURN
```

Following what I believe to be the best method in writing programs we will start with the graphics section, and work our way forward to the beginning of the program. For the dice we need just six characters (no blank). The same CHR$ numbers will be used as in the last game, so there will be little confusion, but of course there will be no number 240, for the blank is not now needed. Therefore the first line to enter will be SYMBOL AFTER 241.

```
5020 SYMBOL AFTER 241
5030 SYMBOL 241,255,255,255,231,255,
255,255,255:REM DIE 1
5040 SYMBOL 242,255,159,255,255,255,
249,255,255:REM DIE 2
5050 SYMBOL 243,255,159,255,231,255,
249,255,255:REM DIE 3
5060 SYMBOL 244,255,153,255,255,255,
153,255,255:REM DIE 4
5070 SYMBOL 245,255,153,255,231,255,
153,255,255:REM DIE 5
5080 SYMBOL 246,255,153,255,153,255,
153,255,255:REM DIE 6
```

In order to see whether or not those graphic characters have been entered correctly we will use two lines in section 3000. One of them will remain there, and the other (line 3030) can be deleted when we have seen the result.

```
3020 A$=CHR$(241)+CHR$(242)+CHR$(243
)+CHR$(244)+CHR$(245)+CHR$(246)
3030 LOCATE 10,10:PRINT A$
```

When the program is now RUN a line of dice (from 1 to 6) will appear near the middle of the screen. Line 3030 can now be deleted, and we will proceed with section 4000, the Game screen.

For making a lay-out such as this it is essential to use a plotting board, even if it is only roughly drawn on paper. If you have not read the previous chapter, then do so now. Pay attention to the details of how to make such a board, and make one now. Even though it is possible to just enter the listings I shall give, it will be to your advantage, if you wish to learn programming, to go through all the motions of planning this game.

A representation of a table top should be shown on the screen, on which to roll the dice. It can consist of a solid black surround to the table top. This will occupy a space from 1 to 19 (X), by 7 to 17 (Y). Pencil this in lightly to form a surround at rows 7 and 17, and down column 19. This encloses the dice rolling positions.

```
4020 FOR X=1 TO 18:LOCATE X,7:PRINT
CHR$(143):NEXT X
4030 FOR X=1 TO 18:LOCATE X,17:PRINT
 CHR$(143):NEXT X
4040 FOR Y=7 TO 17:LOCATE 19,Y:PRINT
 CHR$(143):NEXT Y
4050 PEN 2:FOR X=1 TO 18:FOR Y=8 TO
16:LOCATE X,Y:PRINT CHR$(207):NEXT Y
:NEXT X:PEN 1
```

When this is RUN there will be displayed at the left middle of the screen an area of bluish green surrounded by a solid black line, which we shall call the table top. Notice the way in which line 4050 creates the patch of blue, using PEN 2 at its commencement, and PEN 1 to finish. The dice will roll on this table, but more of that later. CHR$(143) and CHR$(207) are standard Amstrad characters.

We must provide a title for the Game screen: Chuck-a-Luck. Pencil that in on the plotting board from 13 to 28 (X), on row 2 (Y). That is about central. With a Cross at each end, and printed in red (PEN 3) that should be satisfactory.

A line of instruction is then needed: 'Place your bet on a number (1–6)', and that can be pencilled in on row 4 (Y), starting at column 5 (X). The statement of money available is left. Put 'You have £ ' at column 23(X) and row 6(Y). Then comes the Score position at 24,9; the conversion of dice to their equivalent numbers (in order to pinpoint those that are equal to the number on which you bet) at 21,11: 21,13: and 21,15. At the bottom a statement of the odds you have been offered: at 10,19:10,20: and 10,21.

Lightly pencil these on the plotting board, so that, later on when the program is being written you will know where to place the various numbers that need to be entered. This matter is emphasized because eventually, when writing your own programs, you will realise that it is essential.

```
4060 PEN 3:LOCATE 13,2:PRINT "* CHUC
K-A-LUCK *":PEN 1
4070 LOCATE 5,4:PRINT "PLACE YOUR BE
T ON A NUMBER (1-6)"
4080 LOCATE 23,6:PRINT "YOU HAVE £
 "
4090 LOCATE 24,9:PRINT "SCORE"
4100 LOCATE 21,11:PRINT "DICE 1:"
4110 LOCATE 21,13:PRINT "DICE 2:"
4120 LOCATE 21,15:PRINT "DICE 3:"
4130 LOCATE 10,19:PRINT "1.CORRECT:
MONEY BACK"
4140 LOCATE 10,20:PRINT "2.CORRECT:
DOUBLE BACK"
4150 LOCATE 10,21:PRINT "3.CORRECT:
TREBLE BACK"
```

Notice the PEN 3 (red) in line 4060, and the way PEN 1 is used at the end to revert to black. This completes the section at 4000, and we move to 3000, where the dice are rolled on to the table top. This entails some serious thinking, for it is necessary to give each die the appearance of rolling along the table to its resting position. Fortunately this is easy to do by arranging that a space is printed immediately behind each die as it moves on; but it has to be in PEN 2 (blue) because the table top is blue.

In addition we must see that each die is different: that it is produced randomly. For this we will borrow the RND routine used in the last game but with some additions.

```
3030 B$=A$
3040 C$=A$
3050 RANDOMIZE TIME
3060 O=INT(RND*6)+1
3070 P=INT(RND*6)+1
3080 Q=INT(RND*6)+1
3090 R$=MID$(A$,O,1)
3100 S$=MID$(B$,P,1)
3110 T$=MID$(C$,Q,1)
```

Notice that A$ is made to equal a string of six dice (1 to 6), and that B$ and C$ are given the same value. O, P, and Q are made to equal the RND number, and then come lines 3090 to 3110.

These three lines need studying, for they separate out one of the string of six dice, and do this randomly. R$ is made to equal a single, randomly chosen die using O. S$ uses one from P, and T$ uses one from Q. The use of three sets in this way ensures that the computer produces each one separately, and this prevents the random factor from becoming blurred (see listing on p.31).

When these lines have been entered RUN the program. Each die will roll in a leisurely way, which may seem to be too slow. Possibly it is. However, this slowness has a cause, and a little study of one of these lines will do more to teach you about the way a computer performs its task than would thousands of words.

```
3130 FOR X=1 TO 16:LOCATE X,10:PRINT
 R$:FOR Z=1 TO 15:PEN 2:LOCATE Z,10:
PRINT CHR$(207):NEXT Z:PEN 1:NEXT X:
SOUND 1,60,20
3140 FOR X=1 TO 14:LOCATE X,12:PRINT
 S$:FOR Z=1 TO 13:PEN 2:LOCATE Z,12:
PRINT CHR$(207):NEXT Z:PEN 1:NEXT X:
SOUND 1,55,20
3150 FOR X=1 TO 15:LOCATE X,14:PRINT
 T$:FOR Z=1 TO 14:PEN 2:LOCATE Z,14:
PRINT CHR$(207):NEXT Z:PEN 1:NEXT X:
SOUND 1,50,20
```

Line 3130, for instance, consists of a Z FOR/NEXT loop enclosed by an X loop. It is that Z loop (printing a blue character (CHR$(207)) behind the die as it rolls that slows it up. Between each printing of the die the computer has to go into the blue mode (PEN 2), print CHR$(207), and go back into the black mode (PEN 1) to print the next die.

If line 4050 (the one that prints the blue table top) had a REM inserted into it, so that it didn't print; and if line 3130 had the Z loop entirely removed, and a space (" ") placed directly in front of R$, then the die would roll so fast that it would be a mere flicker that comes to an abrupt stop at its final position.

To really master the intricacies of such moves it is advisable to experiment in this way. A few moments pressing keys and EDITing will teach you rapidly, but be sure to record the program on tape first. If you decide to dispense with the blue colour, so that the dice will move faster, it is up to you. There are many more ways to roll the dice in games like this, but this is the one I have chosen because it has so many points that illustrate a number of features.

Notice the way SOUND is used at the end of each throw. Of the three numbers after SOUND, the middle one controls the pitch of the note. It can be altered to suit yourself. The final number affects the length of time it sounds but this is affected by what the computer does next.

Section 3000 is now complete, and we must move on to section 2000, in which the dice are scanned, and transformed into numbers that are printed after Dice 1, Dice 2, and Dice 3. They are also given a variable letter (K, L, and M), under which letter they will be remembered by the computer when we need to use them during the game.

```
2020 IF R$=MID$(A$,1,1) THEN K=1:LOC
ATE 28,11:PRINT K
2030 IF R$=MID$(A$,2,1) THEN K=2:LOC
ATE 28,11:PRINT K
2040 IF R$=MID$(A$,3,1) THEN K=3:LOC
ATE 28,11:PRINT K
2050 IF R$=MID$(A$,4,1) THEN K=4:LOC
ATE 28,11:PRINT K
2060 IF R$=MID$(A$,5,1) THEN K=5:LOC
ATE 28,11:PRINT K
2070 IF R$=MID$(A$,6,1) THEN K=6:LOC
ATE 28,11:PRINT K
```

These lines are tedious to enter, but if line 2020 is compared with 2030 it will be seen that the only differences are to the line number, to the 1 after A$, and to the 1 after K=. It is my practice to type one line properly, check it thoroughly, and then make the necessary alterations to form each succeeding line. That is quicker and less tedious.

If those six lines have been entered correctly, and the program is now RUN, a number should appear at the DICE 1 position just before the Break line appears at the bottom. That number will be the same as the top die if you have entered everything correctly.

There are twelve more lines to enter in section 2000. These have other differences than the first six lines, for they are for different dice. In section 2000, therefore, there are three sets of six lines each (see listing top of p.33).

With the completion of section 2000 the preliminary clothing of the original skeleton is nearly complete. RUN it now, and see that the program produces the three dice correctly, and that the correct numbers are produced for each of the three dice.

```
2100 IF S$=MID$(B$,1,1) THEN L=1:LOC
ATE 28,13:PRINT L
2110 IF S$=MID$(B$,2,1) THEN L=2:LOC
ATE 28,13:PRINT L
2120 IF S$=MID$(B$,3,1) THEN L=3:LOC
ATE 28,13:PRINT L
2130 IF S$=MID$(B$,4,1) THEN L=4:LOC
ATE 28,13:PRINT L
2140 IF S$=MID$(B$,5,1) THEN L=5:LOC
ATE 28,13:PRINT L
2150 IF S$=MID$(B$,6,1) THEN L=6:LOC
ATE 28,13:PRINT L
2200 IF T$=MID$(C$,1,1) THEN M=1:LOC
ATE 28,15:PRINT M
2210 IF T$=MID$(C$,2,1) THEN M=2:LOC
ATE 28,15:PRINT M
2220 IF T$=MID$(C$,3,1) THEN M=3:LOC
ATE 28,15:PRINT M
2230 IF T$=MID$(C$,4,1) THEN M=4:LOC
ATE 28,15:PRINT M
2240 IF T$=MID$(C$,5,1) THEN M=5:LOC
ATE 28,15:PRINT M
2250 IF T$=MID$(C$,6,1) THEN M=6:LOC
ATE 28,15:PRINT M
```

We go to 1000 now, and because there are a lot of controls in this section it will be well to know the purpose and position of all the 'Variables' used so far.

A$, B$, and C$ are all equal to the six dice used. R$, S$, and T$ are each equal to a randomly selected die. O, P, and Q are the letters given to the RND functions of A$, B$, and C$ respectively. K, L, and M are given to the numerical equivalents of the thrown dice at the three positions.

We shall have to make more variables: JX to represent the money at location 33,6; A to represent the money you bet (which will have to be deducted from JX), and will go to 20,17; while B is to be the number you bet on, and that goes to 37,17. W represents the Score, and goes at 30,9. These will be enough to start off with. An INPUT comes first, at 2,23.

```
1020 JX=50:LOCATE 33,6:PRINT JX
1025 W=0:LOCATE 30,9:PRINT W
1030 LOCATE 2,23:INPUT "PLACE YOUR B
ET:";A
1040 JX=JX-A:LOCATE 33,6:PRINT "    "
:LOCATE 33,6:PRINT JX
1050 LOCATE 20,17:PRINT "YOU BET £";
A;"ON"
1060 LOCATE 2,23:INPUT "ON WHICH NUM
BER(1-6):";B
1070 IF B<1 OR B>6 THEN GOTO 1060
1080 LOCATE 37,17:PRINT B
1090 GOSUB 3000
1100 GOSUB 2000
```

With ‘GOSUB 3000’ at line 60, and ‘GOSUB 2000’ at line 70, each time RUN was entered the dice rolled. Now, because the game is starting, those two lines must be deleted, so that they can come into effect after the money has been bet and the dice number selected. These two GOSUBs must now be put at lines 1090 and 1100.

In line 1020 £50 has been selected as the money available. This can be any amount you wish. It is given the variable JX, and is printed at 33,6.

An INPUT at line 1030 is placed near the screen bottom; ‘Place your bet’, and the bet you place is named A. In line 1040 the bet is deducted from the money available (JX=JX−A), and is printed again at 33,6. In 1050 a line is printed to inform you that: “You bet £ ”;A (the amount you bet) “ON” (and this waits for the answer to the next INPUT).

This is at line 1060, and asks for the number of the die on which you bet. This number is given the variable B. 1070 merely makes sure that any number other than 1 to 6 does not affect the program, and 1080 prints the number (B) at 37,17, so that the sentence is now complete: “You bet £(A) on (B)”. Then come the two GOSUBs that make the dice roll, and print their numbers. Following the rolling of the dice come a number of complex instructions to the computer.

```
1110 IF K=B THEN W=W+1:LOCATE 31,11:
PRINT "*"
1120 IF L=B THEN W=W+1:LOCATE 31,13:
PRINT "*"
1130 IF M=B THEN W=W+1:LOCATE 31,15:
PRINT "*"
1140 LOCATE 30,9:PRINT W
```

At this point in working out this program I first used Z as the variable for the Score (not W as it is now). The result surprised me, for the Score kept appearing as 15 plus whatever score occurred. Searching through the program I found, in line 3130, 3140, and 3150, that Z had been used in a FOR/NEXT loop (as 1 TO 15). This had evidently registered in the computer's 'mind' as a variable. I therefore changed the Z to a W in all the relevant lines in 1000, and thereafter it worked properly.

This is the sort of thing to look for when writing a program. For some reason the computer had registered Z as a variable worth 15; perhaps because of the PEN 2 command, for that is powerful. It points to the moral that if something does not work as it should, then search for a reason.

The lines 1110 to 1140 tell the computer to print a star opposite any number that is the same as the one you bet on, and also to alter the score to correspond. The next few lines move the game on a little.

```
1150 IF W=0 THEN LOCATE 20,17:PRINT
"  YOU LOSE          ":LOCATE 2,23:P
RINT "                        ":GOTO
 1500
1160 IF W=1 THEN LOCATE 20,17:PRINT
"  YOU WIN           ":JX=JX+A:LOCAT
E 33,6:PRINT JX:GOTO 1500
1170 IF W=2 THEN LOCATE 20,17:PRINT
"  YOU WIN           ":JX=JX+(A*2):L
OCATE 33,6:PRINT JX:GOTO 1500
```

```
1180 IF W=3 THEN LOCATE 20,17:PRINT
"  YOU WIN           ":JX=JX+(A*3):L
OCATE 33,6:PRINT JX:GOTO 1500
1500 IF JX=0 THEN LOCATE 2,23:PRINT
"YOU HAVE NO MONEY LEFT.":FOR X=1 TO
 2000:NEXT X:GOTO 1900
1510 FOR N=100 TO 50 STEP -5:SOUND 1
,N,10:NEXT N
1520 LOCATE 2,23:PRINT "
             ":LOCATE 2,23:INPUT "AN
OTHER GO (Y/N)";K$
1530 IF K$="N" THEN GOTO 1900
1540 IF K$="Y" THEN GOTO 1550
1550 W=0:LOCATE 30,9:PRINT W:LOCATE
2,23:PRINT "
  "

1560 PEN 2:LOCATE 16,10:PRINT CHR$(2
07):LOCATE 14,12:PRINT CHR$(207):LOC
ATE 15,14:PRINT CHR$(207):PEN 1
1570 LOCATE 29,11:PRINT "    ":LOCAT
E 29,13:PRINT "    ":LOCATE 29,15:PR
INT "    ":GOTO 1025
1900 CLS:LOCATE 6,12:PRINT "THANK YO
U FOR PLAYING":FOR N=100 TO 50 STEP-
5:SOUND 1,N,20:NEXT N:
1910 LOCATE 4,18:PRINT "FOR ANOTHER
GAME ENTER ESC TWICE":LOCATE 4,20:PR
INT "THEN ENTER RUN FOR ANOTHER GAME
."

1920 GOTO 1920
```

Lines 1150 to 1180 tell the computer what to do whether the score is 0, 1, 2, or 3, and increases (or not) the money available in each case. Each line also sends the computer to line 1500, where there are more instructions leading to a return to 1025, (incorporating a tidy-up procedure) or to 1900.

Line 1900 thanks you for playing, sounds a tune, and then tells you to enter RUN for another game. 1999 RETURN is not now needed, so is deleted.

These last lines of the program have been explained only briefly, but to understand them properly study listings on pages 34, 35 and 36 together. Follow through the instructions given in each case, and write down on a piece of paper all the variables, and what they stand for. If you do this you will be well on the way to becoming proficient at programming.

When the complete program is safely taped, and checked with CAT (see the User's Instruction book, Ch.8, p.5), then try experimenting with any part of it that you do not understand completely. Only in this way is it possible to really master the art of programming in BASIC.

## Chapter Four

# A DRAUGHT BOARD AND GAME

As a change from dominoes and dice I decided to make this game a simple one of Draughts for two players. Games can be written in which the computer takes a part, but these can become very complicated, and not easy to understand. The intention of this book is to help beginners write games for themselves in as simple a way as possible, so simple examples are given.

Once again we must first produce a skeleton plot of the game: a section each for graphics; screen layout and rules; black moves; and white moves. Graphics will be at 5000; the screen at 4000; black moves at 1000; and white moves at 2000. The first listing is:

```
10 REM DRAUGHTS
20 REM ********
30 BORDER 18:INK 1,0:INK 0,26:CLS
40 GOSUB 5000
50 GOSUB 4000
1000 REM BLACK MOVES
1010 REM ***********
1999 LOCATE 2,22:STOP
2000 REM WHITE MOVES
2010 REM ***********
2999 STOP
4000 REM SCREEN
4010 REM ******
4999 RETURN
5000 REM GRAPHICS
5010 REM ********
5999 RETURN
```

Some of the sections are short, but they are separated well. The graphic characters will be tackled first, and there are only four of these: a black and a white piece; and a black and a white crown. You should by now be sufficiently proficient to design your own, but I will give those I designed for they compensate for the fact that the Amstrad pixels are not quite so wide as they are high: the 8 x 8 pixels produce an oblong; higher than it is wide (not a square). Remember this when designing your own characters; it often pays to have a blank bottom line for the eighth number.

```
5020 SYMBOL AFTER 240
5030 SYMBOL 240,60,66,129,129,129,66
,60,0:REM O
5040 SYMBOL 241,60,126,255,255,255,1
26,60,0:REM *
5050 SYMBOL 242,165,255,129,129,129,
129,66,60:REM WHITE CROWN
5060 SYMBOL 243,165,255,255,255,255,
255,126,60:REM BLACK CROWN
```

That completes section 5000, and now the screen must be laid out. That will require a lot more thought if we want a reasonably sized board on the screen, and in this respect the more thoughtful programmer will score over one who is always in a hurry. So many board games of this sort are allowed only one character space for each square of the board. This is far too small, regardless of how well the game plays.

A board with four character spaces (two above two) for each square will be about right for size. However, we want our black and white pieces to occupy one full character space, and they can only be put into a corner of the 2 above 2 square. We need one character space to be positioned centrally in the middle of each draught board square.

To illustrate what is meant enter the following list. It is put at line 9000, well away from the program, so can be used separately by RUN 9000 (or LIST 9000) as is necessary.

```
9000 REM TEST
9005 CLS
9010 FOR Y=104 TO 360 STEP 32:ORIGIN
 40,Y:DRAW 256,0:NEXT Y
```

```
9020 FOR X=40 TO 296 STEP 32:ORIGIN
X,104:DRAW 0,256:NEXT X
9030 FOR A=0 TO 240 STEP 8:FOR Y=112
 TO 360 STEP 16:ORIGIN 0,Y:PLOT 48+A
,0:NEXT Y:NEXT A
9040 FOR A=0 TO 240 STEP 8:FOR X=48
TO 296 STEP 16:ORIGIN X,56:PLOT 0,48
+A:NEXT X:NEXT A
9050 LOCATE 4,2:PRINT "A B C D E F G
 H":LOCATE 2,4:PRINT "A":PRINT:PRINT
 " B":PRINT:PRINT " C":PRINT:PRINT "
 D":PRINT:PRINT " E":PRINT:PRINT " F
":PRINT:PRINT " G":PRINT:PRINT " H"
9060 LOCATE 6,6:PRINT CHR$(240):LOCA
TE 12,10:PRINT CHR$(241)
9160 LOCATE 2,22:STOP
```

When this is entered correctly, and you RUN 9000, a pattern of lines will be drawn on the screen in the position we shall use for the draught board. Between the solid lines are other, dotted lines that indicate the position of the divisions between character spaces. Location letters have been placed outside the square, and one white piece has been placed at B,B; with a black at E,D. This illustration should help the above explanation. In drawing the draught board we have to place it where the solid lines are: which is midway between some character spaces.

Before leaving that program at 9000 study it carefully. LIST 9000– will produce the full list. The solid lines are drawn in lines 9010 and 9020. Lines 9030 and 9040 produce the dotted lines; take note of how it is done; with an additional FOR/NEXT loop, and with PLOT instead of DRAW. This program should be retained on tape, or you may prefer to write it down in some place where you keep such references.

To proceed with the actual draught board. Using my plotting board I drew it exactly as it is used in program 9000, but we will only require four lines to surround it, for the black squares detail the rest sufficiently.

```
4015 CLS
4020 ORIGIN 40,360:DRAW 256,0:ORIGIN
 40,104:DRAW 256,0:ORIGIN 40,104:DRA
W 0,256:ORIGIN 296,104:DRAW 0,256
4040 FOR X=72 TO 264 STEP 64:FOR Y=3
28 TO 360 STEP 2:ORIGIN X,Y:DRAW 32,
0:NEXT Y:NEXT X
4050 FOR X=40 TO 232 STEP 64:FOR Y=2
96 TO 328 STEP 2:ORIGIN X,Y:DRAW 32,
0:NEXT Y:NEXT X
4060 FOR X=72 TO 264 STEP 64:FOR Y=2
64 TO 296 STEP 2:ORIGIN X,Y:DRAW 32,
0:NEXT Y:NEXT X
4070 FOR X=40 TO 232 STEP 64:FOR Y=2
32 TO 264 STEP 2:ORIGIN X,Y:DRAW 32,
0:NEXT Y:NEXT X
4080 FOR X=72 TO 264 STEP 64:FOR Y=2
00 TO 232 STEP 2:ORIGIN X,Y:DRAW 32,
0:NEXT Y:NEXT X
4090 FOR X=40 TO 232 STEP 64:FOR Y=1
68 TO 200 STEP 2:ORIGIN X,Y:DRAW 32,
0:NEXT Y:NEXT X
4100 FOR X=72 TO 264 STEP 64:FOR Y=1
36 TO 168 STEP 2:ORIGIN X,Y:DRAW 32,
0:NEXT Y:NEXT X
4110 FOR X=40 TO 232 STEP 64:FOR Y=1
04 TO 136 STEP 2:ORIGIN X,Y:DRAW 32,
0:NEXT Y:NEXT X
```

The outside lines are drawn in 4020, and the following eight lines produce the eight rows of black squares. Take note of how they do that, for each line contains a nested loop (Y inside X). The X command contains STEP 64, so that it jumps to the beginning of each of the four squares in turn. The Y command (in line 4110) is just 104 TO 136, so, because the DRAW command is 32,0 (in the X position) it draws horizontal lines 32 pixels long in all the Y positions, until they are used up, and then goes to the next X position.

Our next move is to enter the letters that will locate the different squares on the board. Because we have arranged the board in a position where its top left square encloses the LOCATE/PRINT position at 4,4:, it is now possible to give the letters a number equivalent. This is done in line 4140.

```
4120 LOCATE 4,2:PRINT "A B C D E F G
 H":LOCATE 2,4:PRINT "A":PRINT:PRINT
 " B":PRINT:PRINT " C":PRINT:PRINT "
 D"
4130 LOCATE 2,12:PRINT "E":PRINT:PRI
NT " F":PRINT:PRINT " G":PRINT:PRINT
 " H"
4140 A=4:B=6:C=8:D=10:E=12:F=14:G=16
:H=18
```

If the program is now RUN a board will be drawn, with the location letters at the top and side. Remember always that these have to be entered separately during the play, with the top one first. There is a blank space on the right of the screen, and we shall now fill this with instructions and a title.

```
4150 PEN 3:LOCATE 22,2:PRINT "* * DR
AUGHTS * *":PEN 1
4160 LOCATE 21,4:PRINT "To REMOVE pi
eces:":LOCATE 21,5:PRINT "Press TOP
letter.":LOCATE 21,6:PRINT "then SID
E letter."
4170 PEN 3:LOCATE 21,7:PRINT "S E P
A R A T E L Y":PEN 1
4180 LOCATE 21,9:PRINT "To ENTER pie
ces:":LOCATE 21,10:PRINT "Press TOP
letter.":LOCATE 21,11:PRINT "then SI
DE letter."
4190 PEN 3:LOCATE 21,12:PRINT "S E P
 A R A T E L Y":PEN 1
4200 LOCATE 21,16:PRINT "Answer the
```

```
INPUT":LOCATE 21,17:PRINT "questions
"
4210 PEN 3:LOCATE 21,19:PRINT "C A R
 E F U L L Y":PEN 1
```

These are straightforward entries of text at given locations. Some lines, however, are in red (PEN 3). Notice the way in which the change of colour is confined to its own line (lines 4150, 4170, 4190, and 4210). Used in this way, with PEN 1 at the end of each line to switch it back to black, there is no chance that other lines will be printed red.

The only part of section 4000 that remains is to place the black and white pieces in position, and the lines for doing this are brief.

```
4220 FOR X=A TO G STEP 4:FOR Y=A TO
C STEP 4:LOCATE X,Y:PRINT CHR$(240):
NEXT Y:NEXT X
4230 FOR X=B TO H STEP 4:Y=B:LOCATE
X,Y:PRINT CHR$(240):NEXT X
4240 FOR X=B TO H STEP 4:FOR Y=F TO
H STEP 4:LOCATE X,Y:PRINT CHR$(241):
NEXT Y:NEXT X
4250 FOR X=A TO G STEP 4:Y=G:LOCATE
X,Y:PRINT CHR$(241):NEXT X
```

Pay particular attention to the way the computer deals with line 4220 (and the others in this list). It reads like this, with the computer's 'thoughts' in brackets: FOR (that means a repeat is coming) X=A (that means 4) TO G (that means TO 16) STEP 4 (that means at 4, 8, 12, and 16): FOR (another repeat) Y=A (that means 4) TO C (that means 8) STEP 4 (that means at 4 & 8): LOCATE X,Y: (I've done that) PRINT CHR$(240): (that's easy, PRINT it at 4,4): NEXT Y (I must go back and PRINT it at 8,4) NEXT Y (back to 12,4) NEXT Y (back to 16,4) NEXT Y (that's finished) NEXT X (go back and repeat it all starting at 8,4: at 8,8: at 8,12: at 8,16) NEXT X (I.ve finished that. Next line Please).

Of course the computer doesn't think (or does it?). In any case that is how the various numbers and letters are interpreted. This is emphasized to bring out one point: be careful with your own thinking when giving a computer instructions. At my first entering of that line I thoughtlessly entered 'FOR X=A TO C STEP 2'. I was thinking, wrongly as it turned out, that the computer would know I meant alternate letters. I was wrong. It had already changed the letters into numbers when it met STEP 2. Therefore it printed the pieces on every white and black square along that line. Remember that a computer is simple-minded, and anxious to do exactly as it is told. It interprets very literally all you enter, so be careful how you phrase each line.

This brings up another point. The Amstrad prints its 'Break Ready' notice at a position on the left of the screen just after the last command it has executed. Therefore, whenever you put a STOP line in a program, word it as I have for line 1999, that is: LOCATE 2,22: STOP. This ensures that after it has printed anything, and is finished, its 'Break Ready' notice will be in the least annoying place. If you use 2,23: STOP, it will 'nudge' the whole program up one space to get its message in, for the message takes up three lines, and starts after 2,23.

The game screen (section 4000) is now complete, with the black and white pieces located in their squares ready for playing a game. Therefore we shall move to line 1000 for the first black move. The simplest way to make this move is to enter the location from which the black is to move, and follow it with the location to which it has to go.

For this game, which consists mainly of moves, I have decided that the best INPUT routine to use for each move is with INKEY$, for then the letters of the location are pressed, and the ENTER key is not used. One decides, for instance, that black is to move from DF to EE. The computer's speed of execution is such that with normal key pressing each question will be presented on the screen in turn as fast as you can key them in.

There will have to be a separate routine at 3000 to enable the computer to recognise the letters entered as representing the location numbers. First, however, we will enter the preliminary moves for black in section 1000.

```
1020 LOCATE 2,22:PRINT "BLACK FROM?(
TOP NO.)?              "
1030 GOSUB 3000
1040 LOCATE 2,22:PRINT "BLACK FROM?(
SIDE NO.)?             "
1050 GOSUB 3100
1060 LOCATE X,Y: PRINT " "
1070 LOCATE 2,22:PRINT "BLACK TO?(TO
P NO.)?                "
1080 GOSUB 3000
1090 LOCATE 2,22:PRINT "BLACK TO?(SI
DE NO.)?               "
1100 GOSUB 3100
```

When these have been entered it will be seen that they contain four GOSUB commands. These must be entered now:

```
3000 J$=INKEY$: IF J$="" THEN GOTO 3
000
3005 IF J$="A" THEN X=4
3010 IF J$="B" THEN X=6
3015 IF J$="C" THEN X=8
3020 IF J$="D" THEN X=10
3025 IF J$="E" THEN X=12
3030 IF J$="F" THEN X=14
3035 IF J$="G" THEN X=16
3040 IF J$="H" THEN X=18
3045 RETURN
3100 K$=INKEY$: IF K$="" THEN GOTO 3
100
3105 IF K$="A" THEN Y=4
3110 IF K$="B" THEN Y=6
3115 IF K$="C" THEN Y=8
3120 IF K$="D" THEN Y=10
3125 IF K$="E" THEN Y=12
```

```
3130 IF K$="F" THEN Y=14
3135 IF K$="G" THEN Y=16
3140 IF K$="H" THEN Y=18
3145 RETURN
```

There are two sub-sections in this long listing, and they have been arranged in such a way that they can be used by either black or white. Therefore we shall make another series of moves (for white) before explaining how section 3000 works.

```
2020 LOCATE 2,22:PRINT "WHITE FROM?(
TOP NO.)?                  "
2030 GOSUB 3000
2040 LOCATE 2,22:PRINT "WHITE FROM?(
SIDE NO.)?                 "
2050 GOSUB 3100
2060 LOCATE X,Y: PRINT " "
2070 LOCATE 2,22:PRINT "WHITE TO?(TO
P NO.)?                    "
2080 GOSUB 3000
2090 LOCATE 2,22:PRINT "WHITE TO?(SI
DE NO.)?                   "
2100 GOSUB 3100
```

Also enter these four lines:

```
1110   LOCATE X,Y:PRINT CHR$(241)
1120   GOTO 2020
2110   LOCATE X,Y:PRINT CHR$(240)
2120   GOTO 1020
```

The white moves are almost identical with those of black. In the sub-routine at 3000 it is fairly easy to see what has been done. Line 3000 tells the computer to wait until a key is pressed: J$=INKEY$: IF J$=“ ” THEN GOTO 3000. Lines 3005 to 3040 tell the computer which X number to use for each of the letters A to H, and line 3045 returns the computer back to the line after the one that sent it to 3000.

Lines 3100 to 3145 are a repetition of 3000 to 3045, but directed at the Y number this time. These two sub-sections are a good demonstration of how the GOSUB/RETURN routine can save lines in a program: without it each of the ten lines would have to be placed in the section at 1000 and 2000 in the places at present occupied by the line GOSUB 3000, or GOSUB 3100. These two sub-sections will be used fairly often during the game. The use of INKEY$ for INPUT also has many advantages, so make a note of how it is done.

With these lists now entered (plus those four extra lines) you will be able to move any piece on the board to a different location, subject to the rule that black and white move alternately. Use RUN now, and check that the moves are operating correctly. If they are not, then you have made an incorrect entry. Lines 3000 and 3100 are particularly sensitive. There is a space between : and IF. No space between the inverted commas (""). A space is between "" and THEN. A space between THEN and GOTO. And a space between GOTO and 3000 (or 3100). It is most important to watch all punctuation and spaces.

Up to now provision has not been made for jumping (possibly over more than one); for the removal of a jumped-over piece; or the exchange of a piece for a crown. There are many complicated ways in which a computer is made to recognise such moves, but in this game such complications are not called for. It will be kept quite simple, and more like the situation would be if you were playing with an actual board and pieces. This entails using a fair amount of repetition, but by using INKEY$ the questions can always be answered with a single key (without ENTER). These next few lines make provision for jumping, and for crowning.

```
1110 LOCATE 2,22:PRINT "IS IT A CROW
N(Y/N)?            "
1115 L$=INKEY$: IF L$<>"Y" AND L$<>"
N" THEN 1115
1120 IF L$="N" THEN LOCATE X,Y:PRINT
 CHR$(241)
```

```
1125 IF L$="Y" THEN LOCATE X,Y:PRINT
 CHR$(243)
1130 LOCATE 2,22:PRINT "HAVE YOU JUM
PED(Y/N)?              "
1140 M$=INKEY$: IF M$<>"Y" AND M$<>"
N" THEN 1140
1150 IF M$="N" THEN GOTO 2020
1160 IF M$="Y" THEN GOTO 1170
1170 LOCATE 2,22:PRINT "REMOVE WHITE
 FROM(TOP NO.)?        "
1180 GOSUB 3000
1190 LOCATE 2,22:PRINT "REMOVE WHITE
 FROM(SIDE NO.)?       "
1200 GOSUB 3100
1210 LOCATE X,Y: PRINT " "
1220 LOCATE 2,22:PRINT "CAN YOU JUMP
 AGAIN(Y/N)?           "
1230 N$=INKEY$: IF N$<>"Y" AND N$<>"
N" THEN 1230
1240 IF N$="N" THEN GOTO 2020
1250 IF N$="Y" THEN GOTO 1170
```

It will be seen that the line 1110 has been changed to introduce the Crown question. The spaces after the question mark in some questions are there to ensure that all the previous question in the space has been completely erased. Because all the questions are located at 2,22;, and some are longer than others, this precaution is necessary.

More lines are needed for a routine that will also cover all aspects of jumping (either single or more) so the original line 1120 was removed. Now, at line 1130, starts a loop that goes to 1250, and comes back to 1170 if more than one jump is made. In the process of removing jumped-over pieces the sub-routine at 3000 and 3100 are used again.

If no jumps have been made the program goes to 2020, and that will be our next move. The lines can all be entered by EDITing from line 1110 onward; changing them to 2110 onward, and changing colours and GOTOs when necessary.

```
2110 LOCATE 2,22:PRINT "IS IT A CROW
N(Y/N)?              "
2115 L$=INKEY$: IF L$<>"Y" AND L$<>"
N" THEN 2115
2120 IF L$="N" THEN LOCATE X,Y:PRINT
 CHR$(240)
2125 IF L$="Y" THEN LOCATE X,Y:PRINT
 CHR$(242)
2130 LOCATE 2,22:PRINT "HAVE YOU JUM
PED(Y/N)             "
2140 M$=INKEY$: IF M$<>"Y" AND M$<>"
N" THEN 2140
2150 IF M$="N" THEN GOTO 1020
2160 IF M$="Y" THEN GOTO 2170
2170 LOCATE 2,22:PRINT "REMOVE BLACK
 FROM(TOP NO.)?      "
2180 GOSUB 3000
2190 LOCATE 2,22:PRINT "REMOVE BLACK
 FROM(SIDE NO.)?     "
2200 GOSUB 3100
2210 LOCATE X,Y: PRINT " "
2220 LOCATE 2,22:PRINT "CAN YOU JUMP
 AGAIN(Y/N)?         "
2230 N$=INKEY$: IF N$<>"Y" AND N$<>"
N" THEN 2230
2240 IF N$="N" THEN GOTO 1020
2250 IF N$="Y" THEN GOTO 2170
```

When this list has been entered the program is complete. It can be tested and played as you would with an ordinary draught board and pieces. The constant questions (Is it a Crown, and Have you jumped?) may annoy you. In that case try modifying the program to ask these questions only when necessary. It can be done in a number of ways, but involves some most convoluted thinking. Make sure you have the program safely taped before you try altering it.

When interested in playing, one is inclined to be impatient, and you may enter a location letter when the question is asking for Y or N. Therefore the lines at 1115, 1140, 1230, 2115, 2140, and 2230 have been modified to prevent a 'Break' if you press the wrong letter. Compare lines 1115 and 3000: you will then see the kind of difference that has been made.

The STOP lines (1999, 2999) may now be removed. If you retain the lines at 9000 onward it will not affect the program at all. In the next chapter we will tackle the game of Reversi.

# Chapter Five

## REVERSI, USING COMPOSITE U.D.G.s

The game of Reversi lends itself well to the methods used in the previous game, for it divides easily into sections: graphics, screen display, black moves, and white moves. The skeleton, or plot, devised for the previous game can be used almost as it stands.

```
10 REM REVERSI
20 REM *******
30 BORDER 2:INK 1,0:INK 0,23:CLS
40 GOSUB 5000
50 GOSUB 4000
1000 REM BLACK MOVES
1010 REM ***********
1999 LOCATE 2,22:STOP
2000 REM WHITE MOVES
2010 REM ***********
2999 LOCATE 2,22:STOP
4000 REM SCREEN
4010 REM ******
4999 RETURN
5000 REM U.D.GRAPHICS
5010 REM ************
5999 RETURN
```

We start with the graphics, and there will be nine of these. In the game of Reversi the jumped-over opponent is replaced by the opposite colour, so there is no need for a replacement of the background. However, the counters are usually bigger than in other games, and occupy four spaces instead of one. Another difference brought about by the larger counters is that each square of the board can enclose four complete character spaces, instead of cutting across some of them, so we can devise counters to fill four character spaces each.

If you are still not sure how to do this the following extra program will soon teach you. It has been written to sit at line 9000 onward, so can be entered and RUN without interfering with the game of Reversi.

```
9000 REM TEST
9005 CLS
9010 FOR Y=48 TO 304 STEP 128:X=112:
ORIGIN X,Y:DRAW 256,0:NEXT Y
9020 FOR X=112 TO 368 STEP 128:Y=48:
ORIGIN X,Y:DRAW 0,256:NEXT X
9030 FOR A=1 TO 256 STEP 6:FOR Y=48
TO 304 STEP 16:X=1:ORIGIN X,Y:PLOT 1
12+A,0:NEXT Y:NEXT A
9040 FOR A=1 TO 272 STEP 4:FOR X=112
 TO 368 STEP 16:Y=1:ORIGIN X,Y:PLOT
0,48+A:NEXT X:NEXT A
9050 LOCATE 8,3:PRINT "1       1":LO
CATE 8,4:PRINT "2631    2631":LOCATE
 8,5:PRINT "8426842184268421"
9060 PRINT CHR$(22)+CHR$(1):LOCATE 8
,7:PRINT "*":LOCATE 8,8:PRINT "* * *
 * * * * *":LOCATE 8,9:PRINT "*   **
******":LOCATE 8,10:PRINT "* **
   * *"
9070 LOCATE 8,11:PRINT "* *
 *":LOCATE 8,12:PRINT "**
 *":LOCATE 8,13:PRINT "**
 *":LOCATE 8,14:PRINT "**
 *"
9080 LOCATE 8,15:PRINT "**
   *":LOCATE 8,16:PRINT "**
   *":LOCATE 8,17:PRINT "**
   *":LOCATE 8,18:PRINT "* *
 *"
9090 LOCATE 8,19:PRINT "*  *
*":LOCATE 8,20:PRINT "* * *********
```

```
*":LOCATE 8,21:PRINT "*":LOCATE 8,22
:PRINT "****************":PRINT CHR$
(22)+CHR$(0)
9100 LOCATE 4.7:PRINT "128":PRINT "
  170":PRINT "   143":PRINT "   176"
9110 LOCATE 4,11:PRINT "160":PRINT "
   192":PRINT "   192":PRINT "   192
"
9120 LOCATE 4,15:PRINT "192":PRINT "
   192":PRINT "   192":PRINT "   160
"
9130 LOCATE 4,19:PRINT "144":PRINT "
   175":PRINT "   128":PRINT "   255
"
9140 LOCATE 25,7:PRINT "  0":LOCATE
25,8:PRINT "170":LOCATE 25.9:PRINT "
240":LOCATE 26,10:PRINT "10"
9150 LOCATE 27,11:PRINT "4":LOCATE 2
7,12:PRINT "2":LOCATE 27,13:PRINT "2
":LOCATE 27,14:PRINT "2"
9160 LOCATE 27,15:PRINT "2":LOCATE 2
7,16:PRINT "2":LOCATE 27,17:PRINT "2
":LOCATE 27,18:PRINT "4"
9170 LOCATE 27.19:PRINT "8":LOCATE 2
5,20:PRINT "250":LOCATE 27.21:PRINT
"0":LOCATE 25,22:PRINT "255"
9200 LOCATE 1,1:STOP
```

This is quite a long sub-program, but if you are still struggling to understand U.D.G.s (User Defined Graphics), then this program, together with some explanations, should make the subject much clearer.

When it is entered correctly use RUN 9000. On the screen will be shown four adjacent sets of 8 x 8 squares; each of the four separated by solid lines, and inside these the 8 x 8 squares are separated by dotted lines. At the top are the values of each column below that number. From the left: 128, 64, 32, 16, 8, 4, 2, 1; and these are repeated for the top right square. These numbers can be remembered easily: from the right-hand side, going left, each succeeding number is double that on its right.

So much for the columns. The ROW numbers (down the left, and down the right) are obtained by counting the values of the 'filled-in' squares in that row. In this block of four characters we will call them: top-left, top-right, bottom-left, and bottom-right. The top-left will be used for explaining the numbers.

The first row has 1 filled-in (*) square under 128, so enter 128. The second row has 4 filled-in squares under 128,32,8,2, so enter 170. The third row has 5 filled-in squares under 128,8,4,2,1, so enter 143. And so it goes on. By now you should be able to add up and fill in the rest; the numbers are given so that you can check that they are correct. Ignore the pattern made on these four squares for the moment (there is an obvious circle, solid lines down the left and bottom edges, as well as some dots outside the circle). This pattern will be explained later.

Having studied this until you really understand what is being done and how to do it, the next decision is how to use the numbers thus produced. Each set of numbers (128 to 192 for the top-left; 0 to 2 for top-right; 192 to 255 bottom-left; and 2 to 255 for bottom-right) are entered in section 5000 (or any other place where you are making U.D.G.s) after the word 'SYMBOL' and its number; and divided from that number and from each other by a comma.

The pattern we have described makes the white counter. The spaces in the circle were filled in then, and new sets of eight numbers produced for the black counter. That should be easy to understand, but the lines down the left and across the bottom as well as the outer dots need more explanation.

I found, when the circle was printed (using the four characters) that they erased the already drawn lines of the board on their immediate left and bottom edge. Therefore those lines were added to the graphic character.

The dots! Well, the background is produced by printing graphic character 240, which is made up with the values of

0,170,0,170,0,170,0,170: alternate rows of spaces and alternate dots. When this counter is printed as a background it is then overdrawn by lines. Then, when our black or white counter is printed on top of this it rubs out some of the background (around the circle). This must be replaced. Therefore dots (fillings-in) are put on the pattern after the circle is drawn (conforming exactly to the way those dots would appear in character 240). They are then counted into the values.

One further point before we leave this section at 9000. The characters on the Amstrad are not so wide as they are high. They are not properly square, but oblong. So for the circle we have made one that is wider than it is high, so that it will print as a proper circle on the screen.

If you have followed this so far you should now find it easy to design your own U.D.G.s, using a pencil and paper ruled into squares. The program at 9000 is worth keeping for reference, for it also shows how certain things are done when programming. Notice the way in which dotted lines are drawn, using PLOT instead of DRAW, and an extra FOR/NEXT loop.

Another interesting point is the way the left-hand column of numbers (for top-left and bottom-left) have been programmed in lines 9100–9130. The colon ( : ) followed by PRINT, tells the computer to put it on the next line down. It would do this at X 1, so three spaces have been inserted inside the inverted commas in front of the number to move it to X 4. That cannot be done on the right-hand side of the drawing because the spaces would be too many, and would erase the drawing along that line.

To return to the main program: we now have nine sets of eight numbers which must be made into graphic characters. The next list does that in section 5000.

```
5020 SYMBOL AFTER 240
5030 SYMBOL 240,0,170,0,170,0,170,0,
170:REM BOARD DOTS
5040 SYMBOL 241,128,170,143,176,160,
192,192,192:REM WHITE.L/TOP
```

```
5050 SYMBOL 242,0,170,240,10,4,2,2,2
:REM WHITE R/TOP
5060 SYMBOL 243,192,192,192,160,144,
175,128,255:REM WHITE L/BOT
5070 SYMBOL 244,2,2,2,4,8,250,0,255:
REM WHITE R/BOT
5080 SYMBOL 245,128,170,143,191,191,
255,255,255:REM BLACK L/TOP
5090 SYMBOL 246,0,170,240,250,252,25
4,254,255:REM BLACK R/TOP
5100 SYMBOL 247,255,255,255,191,159,
175,128,255:REM BLACK L/BOT
5110 SYMBOL 248,254,254,254,254,248,
250,0,255:REM BLACK R/BOT
```

We have used ‘SYMBOL AFTER 240’ once again, for that gives plenty of room. The four parts of the white counter start at CHR$(241) and the four parts of black at 245. In the REM statement at the end of each line is a brief description (white L/Top) etc.

Section 5000 being now complete we move to 4000, which uses CHR$(240) to form the background. Some lines are drawn over that background to form the squares, and there are some letters to locate the square positions.

```
4020 FOR X=4 TO 19:FOR Y=4 TO 19:LOC
ATE X,Y:PRINT CHR$(240):NEXT Y:NEXT
X
4030 FOR X=48 TO 320 STEP 32:Y=96:OR
IGIN X,Y:DRAW 0,254:NEXT X
4040 X=48:FOR Y=96 TO 352 STEP 32:OR
IGIN X,Y:DRAW 254,0:NEXT Y
4050 PRINT CHR$(22)+CHR$(1):LOCATE 4
,3:PRINT "A B C D E F G H":LOCATE 3,
4:PRINT "A":LOCATE 3,6:PRINT "B":LOC
ATE 3,8:PRINT "C"
```

```
4060 LOCATE 3,10:PRINT "D":LOCATE 3,
12:PRINT "E":LOCATE 3,14:PRINT "F":L
OCATE 3,16:PRINT "G":LOCATE 3,18:PRI
NT "H":PRINT CHR$(22)+CHR$(0)
```

The TRANSPARENT mode is used for the letters at the top and side of the board. Use the method of entering suggested previously, until you have checked that they are all entered correctly. That is: omit ‘PRINT CHR$(22)+CHR$(1)’ at first, and then insert it (and also the ending: CHR$(0)) as shown in the list. This is recommended because if an error has been made, and the lines are selected for EDITing while those CHR$ statements are present, they can cause confusion, for they operate while the cursor is moving. It is safer to introduce the TRANSPARENT mode after everything else is checked as correct.

With the board safely entered we now need a title, and some instructions for playing. These are relatively straightforward LOCATE/PRINT statements, but PEN 3 (red) is introduced in alternate paragraphs in order to distinguish them.

```
4070 PEN 3:LOCATE 22,2:PRINT "******
************":LOCATE 22,3:PRINT "* R
 E V E R S I *":LOCATE 22,4:PRINT "**
****************":PEN 1
4080 LOCATE 22,6:PRINT "ONLY Permitt
ed":LOCATE 22,7:PRINT "move is to ju
mp":LOCATE 22,8:PRINT "OVER opponent
:":LOCATE 22,9:PRINT "HORIZONTALLY,"
:LOCATE 22,10:PRINT "DIAGONALLY, or"
:LOCATE 22,11:PRINT "VERTICALLY."
4090 PEN 3:LOCATE 22,12:PRINT "Repla
ce jumped":LOCATE 22,13:PRINT "over
opponent":LOCATE 22,14:PRINT "with o
wn colour.":PEN 1
4100 LOCATE 22,15:PRINT "Black goes
first."
```

```
4110 PEN 3:LOCATE 22,16:PRINT "WINNE
R has most":LOCATE 22,17:PRINT "Piec
es at the end:":LOCATE 22,18:PRINT "
WHEN EITHER":LOCATE 22,19:PRINT "CAN
NOT GO.":PEN 1
```

Because the game of Reversi usually begins with four counters grouped centrally on the board, these must be placed in position as part of the screen lay-out. We will do that now.

```
4120 A$=CHR$(241)+CHR$(242)
4130 B$=CHR$(243)+CHR$(244)
4140 C$=CHR$(245)+CHR$(246)
4150 D$=CHR$(247)+CHR$(248)
4160 LOCATE 10,10:PRINT A$:LOCATE 10
,11:PRINT B$
4170 LOCATE 12,12:PRINT A$:LOCATE 12
,13:PRINT B$
4180 LOCATE 10,12:PRINT C$:LOCATE 10
,13:PRINT D$
4190 LOCATE 12,10:PRINT C$:LOCATE 12
,11:PRINT D$
```

To make the entering of a counter easier the different characters have been grouped into twos: A$ + B$ for white, and C$ + D$ for black. This occurs in lines 4120 to 4150. For the location and printing on the board I used my plotting board to obtain the location numbers (lines 4160 to 4190). When these are all entered use RUN to display the board. It is now complete, and we can proceed to 1000 for the first black move. Five lines will lay down the pattern for all subsequent moves.

```
1020 LOCATE 2,22:PRINT "BLACK TO(TOP
)?              "
1030 GOSUB 3000
1040 LOCATE 2,22:PRINT "BLACK TO(SID
E)?             "
```

```
1050 GOSUB 3200
1060 LOCATE X,Y:PRINT C$:LOCATE X,Y+
1:PRINT D$
```

In line 1020, the first question needing an answer, a lot of spaces have been placed before the final inverted commas. This is repeated in all the following questions because some are longer than others, and the spaces are needed to be sure of erasing all the previous questions in that space. Lines 1030 and 1050 are both GOSUB statements, producing the X,Y: locations for line 1060. Notice how, in order to place the second half of each counter directly below its first half, we use 'X,Y+1' in the second location statement.

```
3000 REM MOVES
3010 REM *****
3020 E$=INKEY$:IF E$="" THEN 3020
3030 IF E$="A" THEN X=4
3040 IF E$="B" THEN X=6
3050 IF E$="C" THEN X=8
3060 IF E$="D" THEN X=10
3070 IF E$="E" THEN X=12
3080 IF E$="F" THEN X=14
3090 IF E$="G" THEN X=16
3100 IF E$="H" THEN X=18
3110 RETURN
3200 F$=INKEY$:IF F$="" THEN 3200
3210 IF F$="A" THEN Y=4
3220 IF F$="B" THEN Y=6
3230 IF F$="C" THEN Y=8
3240 IF F$="D" THEN Y=10
3250 IF F$="E" THEN Y=12
3260 IF F$="F" THEN Y=14
3270 IF F$="G" THEN Y=16
3280 IF F$="H" THEN Y=18
3290 RETURN
```

Lines 3020 and 3200 make the computer wait for an answer. The other lines are self-explanatory: the first section (before 3200) gives the computer its X coordinates, and the second section the Y coordinates. The computer goes through these so fast that its return to the next question is almost too quick to see. No sooner is the top letter entered than the question asking for a side letter appears. We shall continue with black's moves.

```
1070 LOCATE 2,22:PRINT "REMOVE WHITE
(TOP)?             "
1080 GOSUB 3000
1090 LOCATE 2,22:PRINT "REMOVE WHITE
(SIDE)?            "
1100 GOSUB 3200
1110 LOCATE X,Y:PRINT C$:LOCATE X,Y+
1:PRINT D$
1120 PEN 3:LOCATE 2,22:PRINT "CAN YO
U TAKE ANOTHER(Y/N)?     ":PEN 1
1130 G$=INKEY$:IF G$<>"N" AND G$<>"Y
" THEN 1130
1140 IF G$="N" THEN 2000
1150 IF G$="Y" THEN 1070
```

In this section we remove the jumped-over white counter, using the sections at 3000 and 3200 again, and then ask if other white counters can be removed. Because this question is not like those doing the moving, and requires a Y/N answer, it is printed in red so that we shall not miss it. Any number of whites can be removed with a Y answer, for the computer loops back to 1070. When a N is answered, the computer goes to 2000, which is the section for the white moves.

```
2020 LOCATE 2,22:PRINT "WHITE TO(TOP
)?                 "
2030 GOSUB 3000
2040 LOCATE 2,22:PRINT "WHITE TO(SID
E)?                "
2050 GOSUB 3200
```

```
2060 LOCATE X,Y:PRINT A$:LOCATE X,Y+
1:PRINT B$
2070 LOCATE 2,22:PRINT "REMOVE BLACK
(TOP)?                 "
2080 GOSUB 3000
2090 LOCATE 2,22:PRINT "REMOVE BLACK
(SIDE)?                "
2100 GOSUB 3200
2110 LOCATE X,Y:PRINT A$:LOCATE X,Y+
1:PRINT B$
2120 PEN 3:LOCATE 2,22:PRINT "CAN YO
U TAKE ANOTHER(Y/N)?      ":PEN 1
2130 G$=INKEY$:IF G$<>"N" AND G$<>"Y
" THEN 2130
2140 IF G$="N" THEN 1000
2150 IF G$="Y" THEN 2070
```

These lines are similar to the black moves at 1000 onward, and I wrote them into the program by EDITing those at 1020 onward. The line number was given a 2 instead of a 1 at its beginning, and black was changed to white where applicable (and vice versa). In line 2130 'THEN 1130' was changed to 'THEN 2130' and in line 2150 the 'THEN 1070' was changed to 'THEN 2070'.

With these entries completed the game is ready for playing with two players (although I found that playing against oneself can be most interesting, it is easy to change sides and play each colour in turn as if it were one's own colour). There are many complexities to this game that only become apparent after playing it for some time. During the course of a game or two the reason for its title of Reversi becomes understandable, for there are many instances when one colour will dominate the board; and at the next move the position will be reversed. As many as 3, 4, or more can be removed because of one skilful move by the opponent.

For those who have not previously played Reversi it should be explained that the **only** moves allowed are over one (and **only** one) opponent's counter, either horizontally, diagonally, or vertically in any direction (up, down, or to either side). The jumped-over piece is then replaced with one's own colour. On many occasions that replacement will reveal other of the opponent's counters that can be replaced, and so it goes on. An opponent's piece can be replaced by one of your own only as the result of your move and its subsequent removals. The opponent cannot then replace one of yours until he has made a move. It will all become much clearer when you have played the game once or twice.

Save this game on tape, for it can give you many hours of most enjoyable play; but first remove the STOP lines at 1999, and 2999; they are no longer required. The section at 9000 will not affect the game at all if you retain it.

# Chapter Six

# THE ANCIENT GAME OF GO

The game of GO has always fascinated me, so very soon after obtaining my first computer I wrote a program on it for GO. Usually the game is played with black and white pieces (called 'Stones'), which are placed on the intersections of ruled lines. The board has 19 vertical and 19 horizontal lines, making 361 intersections (called 'Points'). Boards are normally about 15 by 17 inches, and so are fairly bulky. Also, the stones (of which there are about 180 of each colour) are easily displaced during play. Because of this I devised a peg-board version of the game, using short, round, black and white pegs as stones.

However, when I wrote a GO program for the Amstrad 464 I found that far more convenient to use, and now I often have it on the screen, either playing a friend, or working out a strategy by myself. In the form used here the computer does not take part; it is written for two players.

The program is relatively simple, for the moves consist of adding or removing pieces. There are only four U.D.G.s used, a screen layout, and the moves, so the skeleton or plot is easy.

```
10 REM * * G O * *
20 REM ***********
30 BORDER 18:INK 1,0:INK 0,26:CLS
40 GOSUB 5000
50 GOSUB 4000
1000 REM BLACK MOVES
1010 REM ***********
1999 LOCATE 2,22:STOP
2000 REM WHITE MOVES
2010 REM ***********
2999 LOCATE 2,22:STOP
4000 REM ******
4999 RETURN
5000 REM U.D.GRAPHICS
5010 REM ************
5999 RETURN
```

The four graphics needed are a black and a white stone, a replacement cross for the intersection (point), and an X for nine points on the board. You should now be able to make these for yourself, but I give the numbers I used because on the black stone and the white stone are projecting points to refill the lines they would otherwise erase.

```
5020 SYMBOL AFTER 240
5030 SYMBOL 240,8,60,66,195,66,66,60
,8:REM WHITE O
5040 SYMBOL 241,8,60,126,255,126,126
,60,8:REM BLACK O
5050 SYMBOL 242,8,8,8,255,8,8,8,8:RE
M CROSS +
5060 SYMBOL 243,8,42,28,255,28,42,8,
8:REM *
```

Notice the REM to each of the lines, so that you will know which CHR$ number represents which character. In section 4000 the drawing of the board (so that the lines intersect centrally on the character spaces), was easy. Providing the board with its location letters was more complicated. However, by drawing it in pencil on a plotting board it became easier. There is a conventional way of numbering a GO board: letters along the top (or bottom) for the columns, and numbers up the left-hand side (1 at the bottom and 19 at the top) for the rows. The numbers running up the left-hand side caused complications in their conversions to Y numbers (which run down the left-hand side), so I have used letters running upward for these. Their conversion was easy; as was the conversion of the top letter, even though they each represented different numbers. The way this was done will be seen when we reach that section, at 3000.

On a GO board there are usually nine marks (like a cross). These are not important for the game as it is normally played, except that they give some idea of location. For instance, if it is intended to play a quarter-sized game (usual for shorter games) the central mark will denote a corner of the board being used; either bounded by A to K and K to T (top-left corner); or K to T and K to T (top-right corner); or A to K and A to K (bottom-left corner); or K to T and A to K (bottom-right corner). When these nine marks are obliterated by a stone during play, they are not replaced if that stone is later removed.

```
4020 FOR X=56 TO 344 STEP 16:Y=72:OR
IGIN X,Y:DRAW 0,294:NEXT X
4030 X=50:FOR Y=72 TO 360 STEP 16:OR
IGIN X,Y:DRAW 294,0:NEXT Y
4040 PEN 3:LOCATE 4,2:PRINT "ABCDEFG
HJKLMNOPQRST"
4050 LOCATE 3,3:PRINT "T":PRINT "  S
":PRINT "  R":PRINT "  Q":PRINT "  P
":PRINT "  O":PRINT "  N":PRINT "  M
":PRINT "  L":PRINT "  K":PRINT "  J
":PRINT "  H"
4060 PRINT "  G":PRINT "  F":PRINT "
  E":PRINT "  D":PRINT "  C":PRINT "
  B":PRINT "  A":PEN 1
```

Notice that the letter I is not used in the location letters: this is to prevent confusion with number 1. Notice also that the lines forming the board are extended slightly towards the letters in order to make location easier. The positioning of these lines (first drawn on the plotting board) is arranged so that they cross each other in the centre of a character space. This ensures correct positioning for the replacing or removal of stones during play.

In lines 4050 and 4060 note how (after the first location is given for the letters downward on the left-hand side) a colon ( : ) followed by PRINT is all that is used. This tells the computer to put it on the next line below; the spaces added between the inverted commas make sure that the letter is printed at its correct distance from the edge of the screen. This was mentioned in a previous chapter, but now there is a slight difference. In this program it is shown that this function will continue when a new line is used; PRINT (without a location) is always just one line lower.

When the program is now RUN it will be seen that there is a blank space on the right side of the screen. This will be filled with a title, some instructions and the scores.

```
4070 PEN 2: FOR X=28 TO 34:FOR Y=3 T
O 5:LOCATE X,Y:PRINT CHR$(207):NEXT
Y:NEXT X:PEN 1
4080 PEN 3: PRINT CHR$(22)+CHR$(1):L
OCATE 30,4:PRINT "G O":PRINT CHR$(22
)+CHR$(0):PEN 1
4090 PEN 3:LOCATE 27,2:PRINT "******
***":LOCATE 27,6:PRINT "*********":F
OR Y=3 TO 5:LOCATE 27,Y:PRINT "*":LO
CATE 35,Y:PRINT "*":NEXT Y:PEN 1
4100 LOCATE 24,8:PRINT "BLACK goes f
irst"
4110 LOCATE 24,10:PRINT "PRESS TOP L
ETTER":LOCATE 24,11:PRINT "& SIDE LE
TTER."
4120 PEN 3:LOCATE 27,13:PRINT "SEPAR
ATELY":PEN 1
4130 LOCATE 24,16:PRINT "BLACK SCORE
:":LOCATE 24,18:PRINT "WHITE SCORE:"

4140 D$=CHR$(243):LOCATE 7,6:PRINT D
$:LOCATE 13,6:PRINT D$:LOCATE 19,6:P
RINT D$
4150 LOCATE 7,12:PRINT D$:LOCATE 13,
12:PRINT D$:LOCATE 19,12:PRINT D$
4160 LOCATE 7,18:PRINT D$:LOCATE 13,
18:PRINT D$:LOCATE 19,18:PRINT D$
```

For the title (because it consists of two letters only) a red surround and a blue background have been arranged. Line 4070 starts with PEN 2 (this is 20, bright cyan), and then two FOR/NEXT loops that locate the area to be made blue. Amstrad character 207 is used for this.

To print the title in red, without spoiling its background, calls for PEN 3, followed by the TRANSPARENT command. If this is entered carefully to ensure that no EDITing is required, then there is no reason to leave the command (CHR$(22)+CHR$(1)) section until after the line is completed.

The printing of the nine marks on the board bring up a point worth mentioning. To avoid writing CHR$(243) a lot of times, D$ is made to equal CHR$(243) in line 4140.

With section 4000 now complete, including the score positions, we can move on to the moves at 1000. These are fairly straightforward, with inputs operated by INKEY$, so that the pressing of the correct key (without ENTER) is all that is needed. However, to establish a routine for the conversion of the location letters to the X or Y numbers, we must have a sub-routine, for it will be needed often. This should be entered first, locating it at section 3000.

```
3000 REM X CONVERSIONS
3010 REM *************
3020 X$=INKEY$:IF X$="" THEN 3020
3030 IF X$="A" THEN X=4
3040 IF X$="B" THEN X=5
3050 IF X$="C" THEN X=6
3060 IF X$="D" THEN X=7
3070 IF X$="E" THEN X=8
3080 IF X$="F" THEN X=9
3090 IF X$="G" THEN X=10
3100 IF X$="H" THEN X=11
3110 IF X$="J" THEN X=12
3120 IF X$="K" THEN X=13
3130 IF X$="L" THEN X=14
3140 IF X$="M" THEN X=15
3150 IF X$="N" THEN X=16
3160 IF X$="O" THEN X=17
3170 IF X$="P" THEN X=18
3180 IF X$="Q" THEN X=19
3190 IF X$="R" THEN X=20
```

```
3200 IF X$="S" THEN X=21
3210 IF X$="T" THEN X=22
3220 RETURN
3300 REM Y CONVERSIONS
3310 REM *************
3320 Y$=INKEY$:IF Y$="" THEN 3320
3330 IF Y$="T" THEN Y=3
3340 IF Y$="S" THEN Y=4
3350 IF Y$="R" THEN Y=5
3360 IF Y$="Q" THEN Y=6
3370 IF Y$="P" THEN Y=7
3380 IF Y$="O" THEN Y=8
3390 IF Y$="N" THEN Y=9
3400 IF Y$="M" THEN Y=10
3410 IF Y$="L" THEN Y=11
3420 IF Y$="K" THEN Y=12
3430 IF Y$="J" THEN Y=13
3440 IF Y$="H" THEN Y=14
3450 IF Y$="G" THEN Y=15
3460 IF Y$="F" THEN Y=16
3470 IF Y$="E" THEN Y=17
3480 IF Y$="D" THEN Y=18
3490 IF Y$="C" THEN Y=19
3500 IF Y$="B" THEN Y=20
3510 IF Y$="A" THEN Y=21
```

This consists of two parts: for X; and for Y; both being fairly long and repetitive. They do, however, make the exact location of the X and Y coordinates very definite, and are worth the extra typing effort. Because the X numbers run from 4 to 22, and the Y numbers from 21 to 3, be very careful that you enter them correctly. With this done we can proceed to the black moves, commencing with some variables for the graphics, and for the scores.

```
1020 A$=CHR$(240):B$=CHR$(241):C$=CH
R$(242):D$=CHR$(243)
1030 VB=0:WB=0:LOCATE 36,16:PRINT VB
:LOCATE 36,18:PRINT WB
1040 LOCATE 2,23:PRINT "BLACK TO(TOP
 LET.)?              "
1050 GOSUB 3000
1060 LOCATE 2,23:PRINT "BLACK TO(SID
E LET.)?             "
1070 GOSUB 3300
1080 LOCATE X,Y:PRINT B$
1090 PEN 3:LOCATE 2,23:PRINT "CAN YO
U TAKE WHITE(Y/N)?":PEN 1
1100 Z$=INKEY$:IF Z$<>"Y" AND Z$<>"N
" THEN 1100
1110 IF Z$="N" THEN 2000
1120 IF Z$="Y" THEN 1130
1130 LOCATE 2,23:PRINT "REMOVE WHITE
(TOP LET.)?          "
1140 GOSUB 3000
1150 LOCATE 2,23:PRINT "REMOVE WHITE
(SIDE LET.)?         "
1160 GOSUB 3300
1170 LOCATE X,Y:PRINT C$:VB=VB+1:LOC
ATE 36,16:PRINT VB
1180 SOUND 1,50,20
1190 GOTO 1090
```

Those lines contain all the black moves. Lines 1040 to 1080 are the moves for locating and printing a black stone on the board, and they include the two GOSUB commands to 3000 and 3300, which establish the locations.

Then comes a question to be answered with Y/N: “Can you take white?”. This appears after every move in the interests of simplicity, and is entered in red to avoid it being missed. If the answer is N then the computer goes to the white moves at 2000. If it is Y it proceeds to introduce the routine for removing the white, using the sub-routines at 3000 and 3300. After the stone is removed line 1190 loops back to 1090, so that further removals can be made or the computer receives a N, and goes on to the white moves.

These lines have been made as simple as possible, and are repeated for the white moves at 2000. You can enter those directly, or, as I did, you can EDIT the lines from 1020 onward, changing the line number and all colours as you proceed. In lines 1100, 1110, 1120, and 1190 there are also numbers to change, while in line 1170 the variable must be changed: VB to WB.

```
2000 REM WHITE MOVES
2010 REM ***********
2020 A$=CHR$(240):B$=CHR$(241):C$=CH
R$(242):D$=CHR$(243)
2040 LOCATE 2,23:PRINT "WHITE TO(TOP
 LET.)?              "
2050 GOSUB 3000
2060 LOCATE 2,23:PRINT "WHITE TO(SID
E LET.)?             "
2070 GOSUB 3300
2080 LOCATE X,Y:PRINT A$
2090 PEN 3:LOCATE 2,23:PRINT "CAN YO
U TAKE BLACK(Y/N)?":PEN 1
2100 Z$=INKEY$:IF Z$<>"Y" AND Z$<>"N
" THEN 2100
2110 IF Z$="N" THEN 1040
2120 IF Z$="Y" THEN 2130
2130 LOCATE 2,23:PRINT "REMOVE BLACK
(TOP LET.)?          "
2140 GOSUB 3000
2150 LOCATE 2,23:PRINT "REMOVE BLACK
(SIDE LET.)?         "
2160 GOSUB 3300
2170 LOCATE X,Y:PRINT C$:WB=WB+1:LOC
ATE 36,18:PRINT WB
2180 SOUND 1,50,20
2190 GOTO 2090
```

With this entered the game is complete. It should be put on tape first, and then tried out. If it doesn't work in any way then check your entries, for there must be an error somewhere.

The scores were mentioned earlier. Those given on the screen are only a part of the score, and a negative part at that. When the game is ended (by mutual consent or because one player cannot find anywhere to safely place a stone) all the points enclosed by black are counted, and are used as black's true score; but only after deducing the number of black stones taken by white as shown by the white score. Similarly, all points enclosed by white, minus black's score of whites taken, are white's true score.

GO is an extremely complex game, although it may not seem so from what you have seen so far, but the full rules are too extensive to be given here. You should obtain (from the library?) a book about GO. The British GO Association, 60 Wantage Road, Reading, Berks, RG3 2SF, will send you details of how to play if, and only if, you enclose a large (10" x 7") stamped, self-addressed envelope with your enquiry. They are most helpful.

To enable you to play a little (while waiting for your book of rules) I include here a few pointers. The aim is to secure, and surround, more territory (points) than your opponent. Black starts by placing a stone anywhere. They play alternately. All stones remain at the same point unless surrounded. To be surrounded is to have an opponent on each of the points around your stone: that is on the points that are on a line with your own; not diagonally. This needs demonstrating, so I have written a program at 9000 to give some examples. You will see it uses some lines taken from 4020 to 4060 that can be EDITed to 9020 to 9060 to give you a start with the list.

```
9000 REM TEST
9010 CLS
9020 FOR X=56 TO 344 STEP 16:Y=72:OR
IGIN X,Y:DRAW 0,294:NEXT X
9030 X=50:FOR Y=72 TO 360 STEP 16:OR
IGIN X,Y:DRAW 294,0:NEXT Y
```

```
9040 PEN 3:LOCATE 4,2:PRINT "ABCDEFG
HJKLMNOPQRST"
9050 LOCATE 3,3:PRINT "T":PRINT "  S
":PRINT "  R":PRINT "  Q":PRINT "  P
":PRINT "  O":PRINT "  N":PRINT "  M
":PRINT "  L":PRINT "  K":PRINT "  J
":PRINT "  H"
9060 PRINT "  G":PRINT "  F":PRINT "
  E":PRINT "  D":PRINT "  C":PRINT "
  B":PRINT "  A":PEN 1
9070 A$=CHR$(240):B$=CHR$(241):D$=CH
R$(243)
9080 LOCATE 5,3:PRINT A$:LOCATE 9,3:
PRINT A$;A$;A$:LOCATE 15,3:PRINT A$;
A$;A$
9090 LOCATE 4,4:PRINT A$;B$;A$:LOCAT
E 8,4:PRINT A$;B$;B$;B$;A$:LOCATE 14
,4:PRINT A$;B$;B$;B$;A$
9100 LOCATE 5,5:PRINT A$:LOCATE 8,5:
PRINT A$;B$;D$;B$;A$:LOCATE 14,5:PRI
NT A$;B$;D$;B$;A$
9110 LOCATE 5,6:PRINT "A":LOCATE 8,6
:PRINT A$;B$;B$;B$;A$:LOCATE 14,6:PR
INT A$;B$;D$;B$;A$
9120 LOCATE 9,7:PRINT A$;A$;A$:LOCAT
E 14,7:PRINT A$;B$;B$;B$;A$
9130 LOCATE 4,8:PRINT A$:LOCATE 10,8
:PRINT "D":LOCATE 15,8:PRINT A$;A$;A
$
9140 LOCATE 4,9:PRINT B$;A$:LOCATE 1
6,9:PRINT "F"
9150 LOCATE 4,10:PRINT A$:LOCATE 4,1
1:PRINT "B":LOCATE 8,11:PRINT A$;A$;
A$;A$;A$
9160 LOCATE 7,12:PRINT A$;B$;B$;B$;B
$;B$;A$
```

```
9170 LOCATE 7,13:PRINT A$;B$;D$;B$;D
$;B$;A$:LOCATE 16,13:PRINT A$;D$;A$
9180 LOCATE 7,14:PRINT A$;B$;B$;B$;B
$;B$;A$:LOCATE 16,14:PRINT D$;B$;D$
9190 LOCATE 8,15:PRINT A$;A$;A$;A$;A
$:LOCATE 16,15:PRINT A$;D$;A$
9200 LOCATE 10,16:PRINT "E":LOCATE 1
7,16:PRINT "C"
9500 LOCATE 2,22:STOP
```

This is a fairly tedious piece of entering, but will be worth the trouble if you want to play GO. RUN 9000 will now show the examples A to F on your screen.

A and B are examples of a properly surrounded stone; but look at example C: that black is NOT surrounded, for the crosses indicate where the whites should have been placed to surround the black. Study that carefully, for it is the fundamental point of the game.

In example D is a block of black stones surrounded correctly by white except for the 'eye' (marked with an X). It is now possible for white to place one stone in that central eye, and thereby take all those blacks, for they are then completely surrounded both inside and out. Therefore, a single 'eye' is no insurance against capture, **it is vulnerable.**

F shows a similar situation to D, but with two adjacent eyes. This also is vulnerable, for white can afford to throw away one stone by placing it at one of the two eyes (marked with XX). Black will respond by placing one stone at the other eye (to surround and remove white), but then the position reverts to a single eye which allows white to capture all the blacks in F.

A completely invulnerable position is shown at E, where two separate eyes have been built up in one block of black. They are shown with an X. This should be your aim with all blocks of your own stones. White is not allowed (by a suicide clause) to place a stone in a single eye unless by doing so the entire block is thereby surrounded, both inside and out. Where two separate eyes exist this is obviously impossible.

With these few simple instructions you will be able to play the game of GO; but there are many finer points of play brought out by a study of the complete rules and examples. If you are really interested, then obtain a book about GO of which there are many about. Ardent followers of this game claim that it has more depth than does Chess.

## Chapter Seven

# A DIFFICULT GAME – CHESS

Complex problems arise when programming a computer for the game of Chess, even though it will be for two players only (without the computer taking part), for the pieces are varied in character, and some are placed on checkered squares. In this version we shall use U.D.Gs to shade what would otherwise be solid black squares, and there will be six black and six white pieces.

The first task is to outline the plot of the program: U.D.G.s in one section; in another the screen layout; white moves; black moves; and a section for various conversions. This last one will be large, and should be placed centrally in the program, after the white and black moves. The various sections must be kept well separated from each other so that they are easier to remember as the program is built up. This also makes the entering much more simple. In a long and complex program every effort must be made to simplify the details.

```
10 REM *********
15 REM * CHESS *
20 REM *********
30 BORDER 2:INK 1,0:INK 0,23:CLS
40 GOSUB 6000
50 GOSUB 5000
1000 REM ***************
1010 REM * WHITE MOVES *
1020 REM ***************
1999 STOP
2000 REM ***************
2010 REM * BLACK MOVES *
2020 REM ***************
2999 STOP
5000 REM **********
5005 REM * SCREEN *
5010 REM **********
```

```
5999 RETURN
6000 REM ****************
6005 REM * U.D.GRAPHICS *
6010 REM ****************
6500 RETURN
```

When this is entered use RUN. The border will be bright blue, the paper a pale cyan, and the text black. Run will also produce an error message, but ignore that. Enter CLS to clear the screen, then LIST and ENTER, and the list will appear black on a pale cyan paper. There is a considerable amount of entering to do, and that is a restful screen to gaze at.

The User Defined Graphics can now be designed. The Chess pieces I have made fill a character space for the Kings and Queens, with relatively smaller spaces for the lesser pieces. The first character has been designed to refill the centre of the shaded squares after a piece has been removed.

Because the shaded squares occupy two half spaces and a full space each in both directions (horizontally and vertically) it has been found necessary to design eight more U.D.G.s for these: four for a corner each, and four for half a space each. These, of course, must use the same formula as that used for the completely shaded U.D.G. This will be explained properly when the screen layout is written.

That makes twenty-one U.D.G.s in all, and these are entered next. Three REM lines have been used for the section title (and this will be done for each section in future); it makes them more easily located when running through a list of the entire program. Each of these U.D.G. lines ends with a REM statement.

```
6020 SYMBOL AFTER 230
6030 SYMBOL 230,0,85,0,85,0,85,0,85:
REM BACKGROUND
6040 SYMBOL 231,0,0,24,60,60,24,60,1
26:REM B.PAWN
6050 SYMBOL 232,0,85,127,62,62,62,12
7,127:REM B.CASTLE
```

```
6060 SYMBOL 233,0,24,52,110,126,60,1
26,126:REM B.BISHOP
6070 SYMBOL 234,0,12,62,126,14,28,62
,127:REM B.KNIGHT
6080 SYMBOL 235,40,170,170,124,56,12
4,254,254:REM B.QUEEN
6090 SYMBOL 236,24,126,219,219,90,12
6,255,255:REM B.KING
6100 SYMBOL 237,0,0,24,36,36,24,36,1
26:REM W.PAWN
6110 SYMBOL 238,0,85,127,34,34,34,99
,127:REM W.CASTLE
6120 SYMBOL 239,0,24,44,82,66,36,102
,126:REM W.BISHOP
6130 SYMBOL 240,0,12,50,114,10,20,34
,127:REM W.KNIGHT
6140 SYMBOL 241,40,170,170,124,40,68
,130,254:REM W.QUEEN
6150 SYMBOL 242,24,126,153,129,90,10
2,153,255:REM W.KING
6160 SYMBOL 243,0,0,0,0,0,5,0,5:REM
T/L SQUARE
6170 SYMBOL 244,0,0,0,0,0,85,0,85:RE
M T/C SQUARE
6180 SYMBOL 245,0,0,0,0,0,80,0,80:RE
M T/R SQUARE
6190 SYMBOL 246,0,5,0,5,0,5,0,5:REM
C/L SQUARE
6200 SYMBOL 247,0,80,0,80,0,80,0,80:
REM C/R SQUARE
6210 SYMBOL 248,0,5,0,5,0,0,0,0:REM
B/L SQUARE
6220 SYMBOL 249,0,85,0,85,0,0,0,0:RE
M B/C SQUARE
6230 SYMBOL 250,0,80,0,80,0,0,0,0:RE
M B/R SQUARE
```

The last entry is 6500 RETURN, which sends the computer back to line 50 GOSUB 5000. This is the screen layout section, and we will follow it there.

One feature of a checkered board game, either Chess or Draughts, is that a single character must be placed centrally in each square. Therefore, if each square is the equivalent of four character spaces in size (and that is a reasonable size for a board game on a computer) the dividing line between the board squares must cut through some character spaces, both horizontally and vertically. This was explained thoroughly in Chapter Four, the Draughts game, so if you have not read that then do so now. The listing on pages 39 and 40 in Chapter Four gives an explanatory program.

I used a plotting board to locate this chess game on the screen by drawing a grid of squares across the area, and then making provision for filling alternate squares with shading. In making the quarter and half squares necessary I used the complete character formula 0,85,0,85,0,85,0,85: that was used for CHR$(230). How the formula was used is shown in lines 6160 to 6230.

```
5020 PEN 3:FOR X=5 TO 17 STEP 4:FOR
Y=5 TO 17 STEP 4:LOCATE X,0+Y:PRINT
CHR$(243);CHR$(244);CHR$(245):NEXT Y
:NEXT X
5030 FOR X=5 TO 17 STEP 4:FOR Y=5 TO
 17 STEP 4:LOCATE X,1+Y:PRINT CHR$(2
46);CHR$(230);CHR$(247):NEXT Y:NEXT
X
5040 FOR X=5 TO 17 STEP 4:FOR Y=5 TO
 17 STEP 4:LOCATE X,2+Y:PRINT CHR$(2
48);CHR$(249);CHR$(250):NEXT Y:NEXT
X
5050 PRINT CHR$(22)+CHR$(1):FOR X=3
TO 15 STEP 4:FOR Y=7 TO 19 STEP 4:LO
CATE X,0+Y:PRINT CHR$(243);CHR$(244)
;CHR$(245):NEXT Y:NEXT X
```

```
5060 FOR X=3 TO 15 STEP 4:FOR Y=7 TO
 19 STEP 4:LOCATE X,1+Y:PRINT CHR$(2
46);CHR$(230);CHR$(247):NEXT Y:NEXT
X
5070 FOR X=3 TO 15 STEP 4:FOR Y=7 TO
 19 STEP 4:LOCATE X,2+Y:PRINT CHR$(2
48);CHR$(249);CHR$(250):NEXT Y:NEXT
X:PRINT CHR$(22)+CHR$(0)
5080 PEN 1:FOR Y=70 TO 328 STEP 32:O
RIGIN 40,Y:DRAW 256,0:NEXT Y
5090 FOR X=40 TO 296 STEP 32:ORIGIN
X,70:DRAW 0,256:NEXT X
```

When these lines have been entered, and the program RUN, it will be seen that the dots are in red (PEN 3). There will be an error message on the screen, but ignore that, unless of course it refers to a mistake you have made when entering the list. You will know this if the checkered board is not completed, with black lines dividing the red squares.

The rest of the entries in this section are fairly easy, for they deal with the placing of numbers and letters around the board; a title for the game; some instructions; a code (which will be explained later); and the placing on the board of the chess pieces.

```
5170 PEN 3:LOCATE 22,2:PRINT "******
***********":LOCATE 22,3:PRINT "* * 
C H E S S * *":LOCATE 22,4:PRINT "**
***************":PEN 1
5180 PEN 3:LOCATE 4,4:PRINT "A B C D
 E F G H"
5190 LOCATE 2,6:PRINT "8":PRINT:PRIN
T " 7":PRINT:PRINT " 6":PRINT:PRINT
" 5":PRINT:PRINT " 4":PRINT:PRINT "
3":PRINT:PRINT " 2":PRINT:PRINT " 1"
:PEN 1
5200 LOCATE 1,2:PRINT "FROM W:":LOCA
TE 1,22:PRINT "FROM B:"
```

```
5210 LOCATE 24,6:PRINT "TOP LETTER &
":LOCATE 24,7:PRINT "SIDE NUMBER."
5220 PEN 3:LOCATE 25,9:PRINT "SEPARA
TELY":PEN 1
5230 PEN 3:LOCATE 25,11:PRINT "* * *
 * *":LOCATE 26,12:PRINT "C O D E":L
OCATE 25,13:PRINT "* * * * *":PEN 1
5240 LOCATE 25,15:PRINT "PAWN....P":
LOCATE 25,16:PRINT "ROOK....R":LOCAT
E 25,17:PRINT "KNIGHT..N"
5250 LOCATE 25,18:PRINT "BISHOP..B":
LOCATE 25,19:PRINT "QUEEN...Q":LOCAT
E 25,20:PRINT "KING....K"
5260 A$=CHR$(231):B$=CHR$(232):C$=CH
R$(234):D$=CHR$(233):E$=CHR$(235):F$
=CHR$(236)
5270 K$=CHR$(237):L$=CHR$(238):M$=CH
R$(240):N$=CHR$(239):O$=CHR$(241):P$
=CHR$(242):Q$=CHR$(230)
5280 LOCATE 4,6:PRINT B$:LOCATE 6,6:
PRINT C$:LOCATE 8,6:PRINT D$:LOCATE
10,6:PRINT E$
5290 LOCATE 12,6:PRINT F$:LOCATE 14,
6:PRINT D$:LOCATE 16,6:PRINT C$:LOCA
TE 18,6:PRINT B$
5300 FOR X=4 TO 18 STEP 2:Y=8:LOCATE
 X,Y:PRINT A$:NEXT X
5310 FOR X=4 TO 18 STEP 2:Y=18:LOCAT
E X,Y:PRINT K$:NEXT X
5320 LOCATE 4,20:PRINT L$:LOCATE 6,2
0:PRINT M$:LOCATE 8,20:PRINT N$:LOCA
TE 10,20:PRINT O$
5330 LOCATE 12,20:PRINT P$:LOCATE 14
,20:PRINT N$:LOCATE 16,20:PRINT M$:L
OCATE 18,20:PRINT L$
```

When this list is completed, add one temporary line: 1500 GOTO 1500. This will prevent an error message spoiling the screen display. Check that everything is in its correct place.

This will be your first chance to check the U.D.G.s also. If there is anything wrong then check all your entries, paying particular attention to the lines that seem to cause the error. To help check the chess pieces, compare the whites with the blacks.

If any shaded squares are wrong then check lines 5020 to 5070, for 5050 to 5070 include the TRANSPARENT command in order to prevent the quarter and half dotted characters from erasing the earlier squares with their blank portions.

The dividing lines are drawn in lines 5080 and 5090. In line 5190 the exact location of the left-hand side numbers are controlled by the single spaces before all the numbers (except 8), and by the double PRINT statement between each number.

In lines 5260 to 5330 the chess pieces have been given variables, which are used to locate the pieces on the board in their starting positions. These variables will stay in the computer's memory, so that they can be used during play.

It will be seen (on the screen display) that letters have been used across the top (giving the columns an X letter notation), and numbers up the side (to give the rows a Y notation). This is the Continental Notation for Chess. If we had attempted to use the English Notation we would have had two sets of numbers for each square: those for white, and those for black.

Remove temporary line 1500, for now we are going to start section 1000 (the white moves), proceeding step by step through the moves, using 'White From' and 'White To'. After the 'From' must come the replacement of the shaded square, or the plain one; and after the 'To' will come the printing of the piece moved.

```
1030 LOCATE 2,24:PRINT "WHITE FROM(T
OP LET.)?              "
1040 GOSUB 3000
```

```
1050 LOCATE 2,24:PRINT "WHITE FROM(S
IDE NO.)?              "
1060 GOSUB 3100
1070 LOCATE X,Y:PRINT " ":GOSUB 3200
```

These few lines are easy, but they contain three subroutines, all of which can be used by either black or white. We will place these at 3000, leaving room at 2000 for the black moves. It will be seen in this section (3000) that 5 is used for the difference in line numbers, instead of 10 as previously. This is because it is useful to allow only 100 each in the line numbering, thereby making the routines sit at lines 3000, 3100, 3200, and so on.

```
3000 REM *****************
3005 REM * X CONVERSIONS *
3010 REM *****************
3015 X$=INKEY$:IF X$="" THEN 3015
3020 IF X$="A" THEN X=4
3025 IF X$="B" THEN X=6
3030 IF X$="C" THEN X=8
3035 IF X$="D" THEN X=10
3040 IF X$="E" THEN X=12
3045 IF X$="F" THEN X=14
3050 IF X$="G" THEN X=16
3055 IF X$="H" THEN X=18
3060 RETURN
3100 REM *****************
3105 REM * Y CONVERSIONS *
3110 REM *****************
3115 Y$=INKEY$:IF Y$="" THEN 3115
3120 IF Y$="1" THEN Y=20
3125 IF Y$="2" THEN Y=18
3130 IF Y$="3" THEN Y=16
3135 IF Y$="4" THEN Y=14
3140 IF Y$="5" THEN Y=12
3145 IF Y$="6" THEN Y=10
```

```
3150 IF Y$="7" THEN Y=8
3155 IF Y$="8" THEN Y=6
3160 RETURN
3200 REM *******************
3205 REM * B/G CONVERSIONS *
3210 REM *******************
3215 PEN 3:IF X+Y=6+6 OR X+Y=10+6 OR
 X+Y=14+6 OR X+Y=18+6 THEN LOCATE X,
Y:PRINT Q$:PEN 1
3220 PEN 3:IF X+Y=4+8 OR X+Y=8+8 OR
X+Y=12+8 OR X+Y=16+8 THEN LOCATE X,Y
:PRINT Q$:PEN 1
3225 PEN 3:IF X+Y=6+10 OR X+Y=10+10
OR X+Y=14+10 OR X+Y=18+10 THEN LOCAT
E X,Y:PRINT Q$:PEN 1
3230 PEN 3:IF X+Y=4+12 OR X+Y=8+12 O
R X+Y=12+12 OR X+Y=16+12 THEN LOCATE
 X,Y:PRINT Q$:PEN 1
3235 PEN 3:IF X+Y=6+14 OR X+Y=10+14
OR X+Y=14+14 OR X+Y=18+14 THEN LOCAT
E X,Y:PRINT Q$:PEN 1
3240 PEN 3:IF X+Y=4+16 OR X+Y=8+16 O
R X+Y=12+16 OR X+Y=16+16 THEN LOCATE
 X,Y:PRINT Q$:PEN 1
3245 PEN 3:IF X+Y=6+18 OR X+Y=10+18
OR X+Y=14+18 OR X+Y=18+18 THEN LOCAT
E X,Y:PRINT Q$:PEN 1
3250 PEN 3:IF X+Y=4+20 OR X+Y=8+20 O
R X+Y=12+20 OR X+Y=16+20 THEN LOCATE
 X,Y:PRINT Q$:PEN 1
3255 RETURN
```

The three sets of conversions in this list are well separated, with easy-to-remember first lines. The X conversions at 3000 convert the letters into their X coordinates. The format used is in order to suit the INKEY$ method of input (just pressing the letter, without the use of ENTER). Line 3015 makes the computer wait until a letter is pressed, but if the wrong letter is pressed it cannot be recalled, so think about each move before you make it.

Line 3060 returns the computer to the line after the one it came from (the GOSUB command), and because it always remembers exactly where it came from the routine can be used in other places as well as here.

The Y conversions are similar in design and effect to the X ones, even though numbers are used instead of letters. This method is used here because of the Continental Notation, and it should advance your knowledge of programming considerably.

Some more difficult conversions are used for printing the backgrounds of the dotted squares. It may have been noticed that (in line 1070) every piece removed is replaced first of all by a blank space (" "). If we did not now print the dotted character on the checkered squares with Q$, the board would soon look ragged. Therefore, after printing the blank the computer is sent to line 3200, where it is told to print Q$ if, and only if, certain conditions prevail. Those conditions are laid down in the third list of conversions (3200–3255). These are fairly involved lines, so take care that all the letters, numbers, ORs, and punctuation marks are entered correctly.

Line 3255 sends the computer back to 1080 where there is a continuation of white's first move: TO (top let.) and TO (side number). For these we will use the original first two sub-routines, but then come back to line 1120, where a fourth sub-routine is invoked, followed by a fifth one in line 1190, and a sixth one in line 1210. We will enter all of the remaining white moves now.

```
1080 PEN 1:LOCATE 2,24:PRINT "WHITE
TO(TOP LET.)?                 "
1090 GOSUB 3000
1100 LOCATE 2,24:PRINT "WHITE TO(SID
E NO.)?                "
1110 GOSUB 3100
```

```
1120 PEN 3:LOCATE 2,24:PRINT "GIVE P
IECE CODE(P,R,N,B,Q,K)   ":PEN 1
1130 GOSUB 3300
1140 PEN 3:LOCATE 2,24:PRINT "IS A P
IECE TAKEN?(Y/N):        ":PEN 1
1150 V$=INKEY$:IF V$="" THEN 1150
1160 IF V$="N" THEN 1200
1170 IF V$="Y" THEN 1180
1180 PEN 3:LOCATE 2,24:PRINT "CODE P
IECE TAKEN(P,R,N,B,Q,K)  ":PEN 1
1190 GOSUB 3400
1200 PEN 3:LOCATE 2,24:PRINT "PROMOT
ION FOR PAWN?(Y/N)       ":PEN 1
1210 GOSUB 3700
```

Before detailing any more explanations it will be best to enter the other sub-routines.

```
3300 REM **********
3305 REM * W CODE *
3310 REM **********
3315 U$=INKEY$:IF U$="" THEN 3315
3320 IF U$="P" THEN LOCATE X,Y:PRINT
 K$
3325 IF U$="R" THEN LOCATE X,Y:PRINT
 L$
3330 IF U$="N" THEN LOCATE X,Y:PRINT
 M$
3335 IF U$="B" THEN LOCATE X,Y:PRINT
 N$
3340 IF U$="Q" THEN LOCATE X,Y:PRINT
 O$
3345 IF U$="K" THEN LOCATE X,Y:PRINT
 P$
3350 RETURN
3400 REM *****************
```

```
3405 REM * PRINT W SCORE *
3410 REM *****************
3415 Y$=INKEY$:IF Y$="" THEN 3415
3420 WS=WS+1
3425 IF Y$="P" THEN LOCATE 7+WS,22:P
RINT A$
3430 IF Y$="R" THEN LOCATE 7+WS,22:P
RINT B$
3435 IF Y$="N" THEN LOCATE 7+WS,22:P
RINT C$
3440 IF Y$="B" THEN LOCATE 7+WS,22:P
RINT D$
3445 IF Y$="Q" THEN LOCATE 7+WS,22:P
RINT E$
3450 IF Y$="K" THEN LOCATE 7+WS,22:P
RINT F$
3455 RETURN
```

In lines 1120 and 1130 is given a routine, including one for the code at 3300, that ensures the correct kind of piece being printed. This is a simple method, and one that will be easy to adapt to any other game you may write.

To provide a record of pieces taken, lines 1140 to 1190 (and the sub-routine at 3400) are used. These, as the game progresses, build up a line of pieces-taken behind the taker's position.

Lines 1200 and 1210 introduce a pawn promotion routine. This is not often needed, but is used after every move throughout the game in the interests of simplicity. Those two lines, with the sub-routine at 3700, are capable of being altered so that they are called into use only when necessary. This is an improvement that you may like to introduce yourself when your grasp of the BASIC language is more complete. At this stage it would be confusing to do so.

```
3500 REM **********
3505 REM * B CODE *
3510 REM **********
```

```
3515 U$=INKEY$:IF U$="" THEN 3515
3520 IF U$="P" THEN LOCATE X,Y:PRINT
 A$
3525 IF U$="R" THEN LOCATE X,Y:PRINT
 B$
3530 IF U$="N" THEN LOCATE X,Y:PRINT
 C$
3535 IF U$="B" THEN LOCATE X,Y:PRINT
 D$
3540 IF U$="Q" THEN LOCATE X,Y:PRINT
 E$
3545 IF U$="K" THEN LOCATE X,Y:PRINT
 F$
3550 RETURN
3600 REM *****************
3605 REM * PRINT B SCORE *
3610 REM *****************
3615 Y$=INKEY$:IF Y$="" THEN 3615
3620 BS=BS+1
3625 IF Y$="P" THEN LOCATE 7+BS,2:PR
INT K$
3630 IF Y$="R" THEN LOCATE 7+BS,2:PR
INT L$
3635 IF Y$="N" THEN LOCATE 7+BS,2:PR
INT M$
3640 IF Y$="B" THEN LOCATE 7+BS,2:PR
INT N$
3645 IF Y$="Q" THEN LOCATE 7+BS,2:PR
INT O$
3650 IF Y$="K" THEN LOCATE 7+BS,2:PR
INT P$
3655 RETURN
3700 REM ************
3705 REM W.PROMOTIONS
3710 REM ************
3715 W$=INKEY$:IF W$="" THEN 3715
```

```
3720 IF W$="N" THEN RETURN
3725 IF W$="Y" THEN 3730
3730 LOCATE 2,24:PRINT "PROMOTE TO(Q
/B/N/R)?         "
3735 T$=INKEY$:IF T$="" THEN 3735
3740 IF T$="Q" THEN 3760
3745 IF T$="B" THEN 3785
3750 IF T$="N" THEN 3810
3755 IF T$="R" THEN 3835
3760 LOCATE 2,24:PRINT "QUEEN TO(TOP
 LET.)?          "
3765 GOSUB 3000
3770 LOCATE 2,24:PRINT "QUEEN TO(SID
E NO.)?          "
3775 GOSUB 3100
3780 LOCATE X,Y: PRINT O$:RETURN
3785 LOCATE 2,24:PRINT "BISHOP TO(TO
P LET.)?         "
3790 GOSUB 3000
3795 LOCATE 2,24:PRINT "BISHOP TO(SI
DE NO.)?         "
3800 GOSUB 3100
3805 LOCATE X,Y: PRINT N$:RETURN
3810 LOCATE 2,24:PRINT "KNIGHT TO(TO
P LET.)?         "
3815 GOSUB 3000
3820 LOCATE 2,24:PRINT "KNIGHT TO(SI
DE NO.)?         "
3825 GOSUB 3100
3830 LOCATE X,Y:PRINT M$:RETURN
3835 LOCATE 2,24:PRINT "ROOK TO(TOP
LET.)?           "
3840 GOSUB 3000
3845 LOCATE 2,24:PRINT "ROOK TO(SIDE
 NO.)?           "
3850 GOSUB 3100
3855 LOCATE X,Y:PRINT L$:RETURN
```

```
4000 REM ************
4005 REM B.PROMOTIONS
4010 REM ************
4015 W$=INKEY$:IF W$="" THEN 4015
4020 IF W$="N" THEN RETURN
4025 IF W$="Y" THEN 4030
4030 LOCATE 2,24:PRINT "PROMOTE TO(Q
/B/N/R)?           "
4035 T$=INKEY$:IF T$="" THEN 4035
4040 IF T$="Q" THEN 4060
4045 IF T$="B" THEN 4085
4050 IF T$="N" THEN 4110
4055 IF T$="R" THEN 4135
4060 LOCATE 2,24:PRINT "QUEEN TO(TOP
 LET.)?             "
4065 GOSUB 3000
4070 LOCATE 2,24:PRINT "QUEEN TO(SID
E NO.)?            "
4075 GOSUB 3100
4080 LOCATE X,Y: PRINT E$:RETURN
4085 LOCATE 2,24:PRINT "BISHOP TO(TO
P LET.)?           "
4090 GOSUB 3000
4095 LOCATE 2,24:PRINT "BISHOP TO(SI
DE NO.)?           "
4100 GOSUB 3100
4105 LOCATE X,Y: PRINT D$:RETURN
4110 LOCATE 2,24:PRINT "KNIGHT TO(TO
P LET.)?           "
4115 GOSUB 3000
4120 LOCATE 2,24:PRINT "KNIGHT TO(SI
DE NO.)?           "
4125 GOSUB 3100
4130 LOCATE X,Y:PRINT C$:RETURN
4135 LOCATE 2,24:PRINT "ROOK TO(TOP
LET.)?             "
```

```
4140 GOSUB 3000
4145 LOCATE 2,24:PRINT "ROOK TO(SIDE
 NO.)?             "
4150 GOSUB 3100
4155 LOCATE X,Y:PRINT B$:RETURN
4160 GOTO 1000
```

The white promotions (at 3700) and the black promotions (at 4000) are given in this list as well as the black code (at 3500). This last code will be used in the black moves starting at 2000, so these should be entered now.

```
2030 LOCATE 2,24:PRINT "BLACK FROM(T
OP LET.)?               "
2040 GOSUB 3000
2050 LOCATE 2,24:PRINT "BLACK FROM(S
IDE NO.)?              "
2060 GOSUB 3100
2070 LOCATE X,Y:PRINT " ":GOSUB 3200
2080 PEN 1:LOCATE 2,24:PRINT "BLACK
TO(TOP LET.)?                "
2090 GOSUB 3000
2100 LOCATE 2,24:PRINT "BLACK TO(SID
E NO.)?               "
2110 GOSUB 3100
2120 PEN 3:LOCATE 2,24:PRINT "GIVE P
IECE CODE(P,R,N,B,Q,K)   ":PEN 1
2130 GOSUB 3500
2140 PEN 3:LOCATE 2,24:PRINT "IS A P
IECE TAKEN?(Y/N):        ":PEN 1
2150 V$=INKEY$:IF V$="" THEN 2150
2160 IF V$="N" THEN 2200
2170 IF V$="Y" THEN 2180
2180 PEN 3:LOCATE 2,24:PRINT "CODE P
IECE TAKEN(P,R,N,B,Q,K)  ":PEN 1
2190 GOSUB 3600
```

```
2200 PEN 3:LOCATE 2,24:PRINT "PROMOT
ION FOR PAWN?(Y/N)        ":PEN 1
2210 GOSUB 4000
2220 GOTO 1000
```

These lines are a repetition of those at 1000 with suitable alterations to the colours mentioned, and to the GOSUB and GOTO references. There is one additional line at the end to send the program back to 1000 for the next white move.

With this entry the game is now complete; but there is one more feature of the game of chess that has not been catered for. This is 'Castling': the exchange of Rook and King on either side and for either colour in certain circumstances. With the knowledge you should have acquired by now it should be quite possible for you to insert a sequence to allow for this. If you use the same method as was used for promoting the pawn, it would require an extra line or two in white's (or black's) moves, and two extra sub-routines after those now finishing at line 4160. This could be an interesting project for those who are seriously attempting to learn programming.

This game of chess has been a long and tedious program to enter, and when put on tape it runs into 5 blocks (about 2 minutes to record or load). Always remember to re-record the whole program each time you add a section. Don't wait until you have checked it, for you can always re-record if any alterations have been made.

In this chapter there have been many new ways in which to achieve one's aim while programming. The use of numbers (left-hand side) as part of the notation for Y, and the clear way that these have been transformed to be used for Y (section 3100) is one of them. As used here that method would not work for numbers over 9, for with a double entry the computer would have moved on before the second half of the number could be pressed.

Of particular interest is the background conversion section at 3200. Notice that X+Y is used instead of X,Y. A somewhat free interpretation of the first part of line 3215 could be: PEN 3: (I must use red ink); IF X+Y= (that is: if location

=) 6+6 (the actual screen coordinates) OR (here's another one) and so on and on. That line deals with the four shaded squares in the top row, and the following lines with following rows. The computer, after printing a blank to erase the piece that has been moved, travels so speedily through the section at 3200 that it is almost impossible to notice the initial erasing with a space before the red dots are printed (with Q$).

In section 3400 (Print W Scores) a 'variable' for the location of the Score is used in line 3420 (WS=WS+1). In the following lines this variable is used to locate each of the pieces taken, and print them according to the code given. This results in a row of pieces collecting as the game progresses. The same thing is done for black in section 3600. The X location is increased by one each time.

Section 3700 (W promotions) has four sub-sections within it, so the procedure is easy to follow. The same applies to the section at 4000 (B promotions).

The method of using quarter, half, or full squares to print red dots forming 32 squares of the board may also be new to you. It calls for the careful use of a plotting board first of all. If you have not got one of these, then the way to make one was described in Chapter Two. If you skipped that chapter then go back now, and follow the instructions for making such a board. Serious programming is almost impossible without the aid of a plotting board of some kind, for it is not easy for anyone to visualise all the points (for PRINT or for PLOT) that are to be used on the screen.

## Chapter Eight

# PONTOON WITH DOMINOES

The game of Pontoon has many followers, and is usually played with a pack of cards. In a computer game, however, the use of dominoes instead of cards is an advantage, for a domino is easily represented on the screen, and the number of them available is unlimited.

For this chapter we will use a game of Domino Pontoon as the base on which to demonstrate a few more points about writing programs for the Amstrad 464. The first of these points is about the use of colour.

Many users of the Amstrad become frustrated by the way in which colours are used. The User Instruction book is not too helpful. In Mode 1 (which we have used exclusively throughout this book) only four colours are available for use on the screen, and those four include the screen colour (Paper). However, those four colours can be selected from most of those available in the list shown on page F3.2 of the User Instruction book. How to select them is shown in line 30 of this new program.

```
10 REM ***************
15 REM DOMINO PONTOON
20 REM ***************
30 INK 0,26:INK 1,0:INK 2,11:INK 3,6
40 BORDER 2:PEN 0::PEN 1:CLS
50 GOSUB 5000
60 GOSUB 4000
```

The words BORDER, PAPER, and PEN should be used as shown, but INK is first used to define the colours selected for the PAPER and PEN commands. This is done in line 30, where INK 0 is made into 26 (using a comma as shown), and becomes bright white; INK 1 becomes 0 (which is black); INK 2 is made to be 11 (which is sky blue); and INK 3 becomes 6 (which is bright red).

I have established a routine of using INK 0 for my PAPER colour, and INK 1 for the main text, with INKs 2 and 3 used as alternatives for display in the program.

So, in line 30, using INK the colours allocated to PAPER and PEN are defined. In line 40 BORDER is given whichever colour I choose, for it does not come under the edict of the INK/PAPER/PEN commands. The second command in line 40 is PAPER, and for that I have used INK 0. For the main text I use INK 1, and I use PEN to call it. Then CLS is used so that the screen is always cleared afresh for the colours.

A little thought will reveal the interesting fact that with a line like 30 in place at the beginning of any program, it is quite easy to alter the colours throughout that program by allocating a different number to the second half of each of the four INK commands. **But**, this cannot be done halfway through a screen routine.

Until they are re-allocated in this way the four colours built into the Amstrad BASIC in Mode 1 are 0=1 (dark blue); 1=24 (bright yellow); 2=20 (bright cyan); and 3=6 (bright red). If, when the program is written, and safely taped, you care to experiment, then alter line 30 to contain the original Amstrad colours (just given). When RUN is now used the effect is totally different. You may prefer it.

With this brief guide to controlling the colours in Mode 1 in your memory, we will proceed with the program. The first move, as usual, is to prepare the U.D.G.s. Because these dominoes will be used vertically instead of horizontally, the data for each line is slightly different than it was for Domino Patience.

```
5000 REM ************
5010 REM U.D.GRAPHICS
5020 REM ************
5030 SYMBOL AFTER 240
5040 SYMBOL 240,255,255,255,255,255,
255,255,255:REM BLANK
5050 SYMBOL 241,255,255,255,231,255,
255,255,255:REM ONE
5060 SYMBOL 242,255,159,255,255,255,
249,255,255:REM TWO
5070 SYMBOL 243,255,159,255,231,255,
249,255,255:REM THREE
5080 SYMBOL 244,255,153,255,255,255,
153,255,255:REM FOUR
5090 SYMBOL 245,255,153,255,231,255,
153,255,255:REM FIVE
5100 SYMBOL 246,255,153,255,153,255,
153,255,255:REM SIX
5110 RETURN
```

No other graphic characters (other than one already in the computer) will be needed, so we can proceed to the section at 4000, to commence designing the screen. Provided there is a plotting board handy the layout of the screen produces very little trouble. Mine was drawn on my Formica board, and from this the locations were read off as required. An area at left centre was reserved for displaying the dominoes; a title was put at the top, with instructions and scores down the right-hand side. The screen is a simple and clear one, and soon entered.

```
4000 REM ***********
4010 REM GAME SCREEN
4020 REM ***********
4040 PEN 3:LOCATE 11,1:PRINT "******
*************":LOCATE 11,2:PRINT "*
DOMINO  PONTOON *":LOCATE 11,3:PRINT
 "*******************":PEN 1
4050 PEN 2:FOR X=4 TO 18:LOCATE X,6:
PRINT CHR$(238):LOCATE X,17:PRINT CH
R$(238):NEXT X:FOR Y=7 TO 16:LOCATE
4,Y:PRINT CHR$(238):LOCATE 18,Y:PRIN
T CHR$(238):NEXT Y:PEN 1
4060 LOCATE 20,5:PRINT "T=TWIST  S=S
TICK":PEN 3:LOCATE 20,6:PRINT "-----
--  -------":PEN 1
```

```
4070 LOCATE 20,10:PRINT "SCORE:":LOC
ATE 20,16:PRINT "SCORE:":PEN 2:LOCAT
E 23,7:PRINT "HOUSE":LOCATE 23,13:PR
INT "YOURS":PEN 1
4080 PEN 3:LOCATE 26,18:PRINT "* GAM
ES *":PEN 1
4090 LOCATE 26,20:PRINT "H-GAMES":LO
CATE 34,20:PRINT "0"
4100 LOCATE 26,22:PRINT "Y-GAMES":LO
CATE 34,22:PRINT "0"
4200 RETURN
```

Notice how PEN 2 (blue) and PEN 3 (red) are used in this section, with PEN 1 at the end of each command to bring the text back to black each time. There is plenty of room left in this section for some sub-routines, but we shall enter those as they are needed.

The bulk of the writing for this game is taken up with moves and conversions, starting at 1000 with placing the dominoes. This will entail producing two randomly chosen dominoes for the move, and the showing of two blank backs for the dominoes dealt to the *House* (the computer).

```
1000 REM **************
1010 REM PLACE DOMINOES
1020 REM **************
1030 GOSUB 4410
1040 LOCATE 6,14:PRINT M$:LOCATE 6,1
4+1:PRINT N$:LOCATE 8,14:PRINT O$:LO
CATE 8,14+1:PRINT P$
1050 LOCATE 6,8:PRINT CHR$(240):LOCA
TE 6,8+1:PRINT CHR$(240):LOCATE 8,8:
PRINT CHR$(240):LOCATE 8,8+1:PRINT C
HR$(240)
1060 GOSUB 3000
1070 LOCATE 20,14:PRINT GH"+"IJ
1080 LOCATE 2,22:PRINT "STICK OR TWI
ST?(S/T)"
1090 X$=INKEY$:IF X$<>"S" AND X$<>"T
" THEN 1090
```

Line 1050 prints the blank dominoes. Notice how the second half of each domino is placed below the first half by adding +1 to the Y coordinate. This could have been done by simply quoting the Y number itself (9 in this instance) but adding the +1 helps to keep one's mind on the sequence of entries. CHR$(240) is the blank, but this is not used in the routine of randomly selecting a domino, because it might keep the numbers so low that more than five dominoes might be asked for (Twist). We have allowed for only five.

Line 1040 prints four domino halves (to make two dominoes) in the position used for your moves, while 1070 prints (as scores) the number of dots on each domino. Two sub-routines have been used, and we shall enter one of these at 4400 now.

```
4400 REM **********
4410 REM * RANDOM *
4420 REM **********
4430 A$=CHR$(241)+CHR$(242)+CHR$(243
)+CHR$(244)+CHR$(245)+CHR$(246)
4440 B$=A$
4450 C$=A$
4460 D$=A$
4470 E$=A$
4480 F$=A$
4490 RANDOMIZE TIME
4500 A=INT(RND*6)+1
4510 B=INT(RND*6)+1
4520 C=INT(RND*6)+1
4530 D=INT(RND*6)+1
4540 E=INT(RND*6)+1
4550 F=INT(RND*6)+1
4560 M$=MID$(A$,A,1)
4570 N$=MID$(B$,B,1)
4580 O$=MID$(C$,C,1)
4590 P$=MID$(D$,D,1)
4600 Q$=MID$(E$,E,1)
4610 R$=MID$(F$,F,1)
```

```
4620 S$=MID$(A$,D,1)
4630 T$=MID$(B$,E,1)
4640 U$=MID$(C$,F,1)
4650 V$=MID$(D$,A,1)
4660 W$=MID$(E$,B,1)
4670 Z$=MID$(F$,C,1)
4680 RETURN
```

This routine, which is carefully written in the simplest and most understandable form, produces randomly chosen dominoes that are (usually) different with each deal. The same method was used in an earlier game. It will be used many times during this game. The blank domino is not used here.

The computer, having printed those two dominoes (line 1040) goes to line 3000, so that the numbers on the dominoes can be changed into ordinary numbers for use in scoring. This routine is essential for the computer to be able to compare numbers, and decide whether or not to use another domino when trying to beat your efforts.

```
3000 REM ****************
3010 REM DOMINO TO NUMBER
3020 REM ****************
3030 IF M$=MID$(A$,1,1) THEN G=1
3040 IF M$=MID$(A$,2,1) THEN G=2
3050 IF M$=MID$(A$,3,1) THEN G=3
3060 IF M$=MID$(A$,4,1) THEN G=4
3070 IF M$=MID$(A$,5,1) THEN G=5
3080 IF M$=MID$(A$,6,1) THEN G=6
3090 IF N$=MID$(B$,1,1) THEN H=1
3100 IF N$=MID$(B$,2,1) THEN H=2
3110 IF N$=MID$(B$,3,1) THEN H=3
3120 IF N$=MID$(B$,4,1) THEN H=4
3130 IF N$=MID$(B$,5,1) THEN H=5
3140 IF N$=MID$(B$,6,1) THEN H=6
3150 GH=G+H
3160 IF O$=MID$(C$,1,1) THEN I=1
```

```
3170 IF O$=MID$(C$,2,1) THEN I=2
3180 IF O$=MID$(C$,3,1) THEN I=3
3190 IF O$=MID$(C$,4,1) THEN I=4
3200 IF O$=MID$(C$,5,1) THEN I=5
3210 IF O$=MID$(C$,6,1) THEN I=6
3220 IF P$=MID$(D$,1,1) THEN J=1
3230 IF P$=MID$(D$,2,1) THEN J=2
3240 IF P$=MID$(D$,3,1) THEN J=3
3250 IF P$=MID$(D$,4,1) THEN J=4
3260 IF P$=MID$(D$,5,1) THEN J=5
3270 IF P$=MID$(D$,6,1) THEN J=6
3280 IJ=I+J
3290 RETURN
```

This somewhat tedious (to enter) list of conversions is only for the first two entered. There will be more lists for each of the additional dominoes used. This method could be abbreviated considerably by using just two strings (M$ and N$) in turn for all the different dominoes. When you have completed the game as it now stands you may like to try your hand (as an exercise) at doing just that. There should be sufficient variation between one deal and the next to ensure different numbers on each deal. However, the purpose of this book, and the way in which it is written, is to ensure that you understand exactly what can be done by even the simplest form of BASIC. You will learn more about how to write programs by such an exercise than by any other method. When trying this don't forget that the 'variables' used to represent the numbers on the dominoes will have to be different for each domino when printing the scores.

Line 3290 sends the computer back to line 1070, where the two numbers (GH & IJ) are printed. Then you are asked to decide whether or not you want another domino to help you to reach the target of 21 (line 1080). Line 1090 makes the computer wait for your answer in the usual manner adopted for an INKEY$ input.

Starting with line 1080 (already entered) is a routine of six lines that is repeated (with detail variations) a number of times. That sequence will be entered next.

```
1100 IF X$="S" THEN 1290
1110 IF X$="T" THEN LOCATE 10,14:PRI
NT Q$:LOCATE 10,14+1:PRINT R$
1120 GOSUB 3300
1130 LOCATE 28,14:PRINT "+"KL
1140 LOCATE 2,22:PRINT "STICK OR TWI
ST?(S/T)"
1150 X$=INKEY$:IF X$<>"S" AND X$<>"T
" THEN 1150
1160 IF X$="S" THEN 1320
1170 IF X$="T" THEN LOCATE 12,14:PRI
NT S$:LOCATE 12,14+1:PRINT T$
1180 GOSUB 3440
1190 LOCATE 32,14:PRINT "+"MN
1200 LOCATE 2,22:PRINT "STICK OR TWI
ST?(S/T)"
1210 X$=INKEY$:IF X$<>"S" AND X$<>"T
" THEN 1210
1220 IF X$="S" THEN 1350
1230 IF X$="T" THEN LOCATE 14,14:PRI
NT U$:LOCATE 14,14+1:PRINT V$
1240 GOSUB 3580
1250 LOCATE 36,14:PRINT "+"OP
1260 LOCATE 2,22:PRINT "STICK OR TWI
ST?(S/T)"
1270 X$=INKEY$:IF X$<>"S" AND X$<>"T
" THEN 1270
1280 IF X$="S" THEN 1380
1290 LOCATE 2,22:PRINT "
         "
1300 YSCORE=GH+IJ:LOCATE 27,16:PRINT
 YSCORE:IF YSCORE>21 THEN 1410
1310 GOTO 2000
1320 LOCATE 2,22:PRINT "
         "
1330 YSCORE=GH+IJ+KL:LOCATE 27,16:PR
INT YSCORE:IF YSCORE>21 THEN 1410
```

```
1340 GOTO 2000
1350 LOCATE 2,22:PRINT "
              "
1360 YSCORE=GH+IJ+KL+MN:LOCATE 27,16
:PRINT YSCORE:IF YSCORE > 21 THEN 14
10
1370 GOTO 2000
1380 LOCATE 2,22:PRINT "
              "
1390 YSCORE=GH+IJ+KL+MN+OP:LOCATE 27
,16:PRINT YSCORE:IF YSCORE>21 THEN 1
410
1400 GOTO 2000
1410 LOCATE 2,22:PRINT "YOU BUST":GO
SUB 4220
1420 LOCATE 2,23:PRINT "ANOTHER GAME
?(Y/N)"
1430 Y$=INKEY$:IF Y$<>"Y" AND Y$<>"N
" THEN 1430
1440 IF Y$="Y" THEN 2580
```

The number of times you press T (for Twist) controls how many of these six-line routines are used. Five should be enough. If (as can happen) you are blessed with a series of small numbers, and don't reach 17, then you will have to be content with what you get. In over 100 tries I have never received less than 20 using all five dominoes.

From 1290 to 1400 are the routines that transfer the moves to the computer (at 2000) but before entering these it is necessary to enter some more sub-routines in the 4000 and 3000 sections.

```
4210 REM ***************
4220 REM * GAMES SCORE *
4230 REM ***************
4240 HG=HG+1
4250 LOCATE 34,20:PRINT HG
4260 RETURN
```

```
4270 YG=YG+1
4280 LOCATE 34,22:PRINT YG
4290 RETURN
```

This small sub-routine (consisting of two parts) prints the number of games that have been won by either competitor. This is a handy way to keep the score.

```
3300 IF Q$=MID$(E$,1,1) THEN K=1
3310 IF Q$=MID$(E$,2,1) THEN K=2
3320 IF Q$=MID$(E$,3,1) THEN K=3
3330 IF Q$=MID$(E$,4,1) THEN K=4
3340 IF Q$=MID$(E$,5,1) THEN K=5
3350 IF Q$=MID$(E$,6,1) THEN K=6
3360 IF R$=MID$(F$,1,1) THEN L=1
3370 IF R$=MID$(F$,2,1) THEN L=2
3380 IF R$=MID$(F$,3,1) THEN L=3
3390 IF R$=MID$(F$,4,1) THEN L=4
3400 IF R$=MID$(F$,5,1) THEN L=5
3410 IF R$=MID$(F$,6,1) THEN L=6
3420 KL=K+L
3430 RETURN
3440 IF S$=MID$(A$,1,1) THEN M=1
3450 IF S$=MID$(A$,2,1) THEN M=2
3460 IF S$=MID$(A$,3,1) THEN M=3
3470 IF S$=MID$(A$,4,1) THEN M=4
3480 IF S$=MID$(A$,5,1) THEN M=5
3490 IF S$=MID$(A$,6,1) THEN M=6
3500 IF T$=MID$(B$,1,1) THEN N=1
3510 IF T$=MID$(B$,2,1) THEN N=2
3520 IF T$=MID$(B$,3,1) THEN N=3
3530 IF T$=MID$(B$,4,1) THEN N=4
3540 IF T$=MID$(B$,5,1) THEN N=5
3550 IF T$=MID$(B$,6,1) THEN N=6
3560 MN=M+N
3570 RETURN
```

```
3580 IF U$=MID$(C$,1,1) THEN O=1
3590 IF U$=MID$(C$,2,1) THEN O=2
3600 IF U$=MID$(C$,3,1) THEN O=3
3610 IF U$=MID$(C$,4,1) THEN O=4
3620 IF U$=MID$(C$,5,1) THEN O=5
3630 IF U$=MID$(C$,6,1) THEN O=6
3640 IF V$=MID$(D$,1,1) THEN P=1
3650 IF V$=MID$(D$,2,1) THEN P=2
3660 IF V$=MID$(D$,3,1) THEN P=3
3670 IF V$=MID$(D$,4,1) THEN P=4
3680 IF V$=MID$(D$,5,1) THEN P=5
3690 IF V$=MID$(D$,6,1) THEN P=6
3700 OP=O+P
3710 RETURN
3720 IF W$=MID$(E$,1,1) THEN Q=1
3730 IF W$=MID$(E$,2,1) THEN Q=2
3740 IF W$=MID$(E$,3,1) THEN Q=3
3750 IF W$=MID$(E$,4,1) THEN Q=4
3760 IF W$=MID$(E$,5,1) THEN Q=5
3770 IF W$=MID$(E$,6,1) THEN Q=6
3780 IF Z$=MID$(F$,1,1) THEN R=1
3790 IF Z$=MID$(F$,2,1) THEN R=2
3800 IF Z$=MID$(F$,3,1) THEN R=3
3810 IF Z$=MID$(F$,4,1) THEN R=4
3820 IF Z$=MID$(F$,5,1) THEN R=5
3830 IF Z$=MID$(F$,6,1) THEN R=6
3840 QR=Q+R
3850 RETURN
```

This long list is tedious to enter, but after a time one begins to work out a routine for entering such sections by EDITing the line above. Because these lines sit on the screen directly above each other in regular columns they are easy to check. When the first one of any six lines is entered the only alteration for the next line is the line number, and the change of a number in two places (rising by 1).

The final list to enter is again a long one, and consists of the moves made by the computer. First of all (line 2030) is GOSUB 4410, so that the computer has fresh access to the randomly chosen dominoes. At line 2040 two are printed in place of the two double blanks put there at the start of the game. Then the moves continue in a similar format to those at 1000; except that instead of the TWIST/STICK options the computer is given a series of options (shown first in lines 2080 to 2110). These will be explained after the list is entered.

```
2000 REM ***********
2010 REM HOUSE MOVES
2020 REM ***********
2030 GOSUB 4410
2040 LOCATE 6,8:PRINT M$:LOCATE 6,8+
1:PRINT N$:LOCATE 8,8:PRINT O$:LOCAT
E 8,8+1:PRINT P$
2050 GOSUB 3000
2060 LOCATE 20,8:PRINT GH"+"IJ
2070 FOR X=1 TO 2000:NEXT X
2080 HSCORE=GH+IJ:IF HSCORE > 17 THE
N LOCATE 27,10:PRINT HSCORE
2090 IF HSCORE > 21 THEN 2530
2100 IF HSCORE < 18 THEN 2120
2110 IF HSCORE > 17 THEN 2350
2120 LOCATE 10,8:PRINT Q$:LOCATE 10,
8+1:PRINT R$
2130 GOSUB 3300
2140 LOCATE 28,8:PRINT "+"KL
2150 FOR X=1 TO 2000:NEXT X
2160 HSCORE=GH+IJ+KL:IF HSCORE > 17
THEN LOCATE 27,10:PRINT HSCORE
2170 IF HSCORE > 21 THEN 2530
2180 IF HSCORE < 18 THEN 2200
2190 IF HSCORE > 17 THEN 2350
2200 LOCATE 12,8:PRINT S$:LOCATE 12,
8+1:PRINT T$
2210 GOSUB 3440
```

```
2220 LOCATE 32,8:PRINT "+"MN
2230 FOR X=1 TO 2000:NEXT X
2240 HSCORE =GH+IJ+KL+MN:IF HSCORE >
 17 THEN LOCATE 27,10:PRINT HSCORE
2250 IF HSCORE>21 THEN 2530
2260 IF HSCORE < 18 THEN 2280
2270 IF HSCORE > 17 THEN 2350
2280 LOCATE 14,8:PRINT U$:LOCATE 14,
8+1:PRINT V$
2290 GOSUB 3580
2300 LOCATE 36,8:PRINT "+"OP
2310 FOR X=1 TO 2000:NEXT X
2320 HSCORE =GH+IJ+KL+MN+OP:IF HSCOR
E > 17 THEN LOCATE 27,10:PRINT HSCOR
E
2330 IF HSCORE > 21 THEN 2530
2340 IF HSCORE > 17 THEN 2350
2350 IF HSCORE = YSCORE THEN 2380
2360 IF HSCORE > YSCORE THEN 2430
2370 IF HSCORE < YSCORE THEN 2480
2380 LOCATE 2,22:PRINT "IT'S A DRAW"
2390 LOCATE 2,23:PRINT "ANOTHER GAME
(Y/N)"
2400 Y$=INKEY$:IF Y$<>"Y" AND Y$<>"N
" THEN 2400
2410 IF Y$="Y" THEN 2580
2420 IF Y$="N" THEN 2420
2430 LOCATE 2,22:PRINT "I WIN":GOSUB
 4220
2440 LOCATE 2,23:PRINT "ANOTHER GAME
?(Y/N)"
2450 Y$=INKEY$:IF Y$<>"Y" AND Y$<>"N
" THEN 2450
2460 IF Y$="Y" THEN 2580
2470 IF Y$="N" THEN 2470
```

```
2480 LOCATE 2,22:PRINT "YOU WIN":GOS
UB 4270
2490 LOCATE 2,23:PRINT "ANOTHER GAME
?(Y/N)"
2500 Y$=INKEY$:IF Y$<>"Y" AND Y$<>"N
" THEN 2500
2510 IF Y$="Y" THEN 2580
2520 IF Y$="N" THEN 2520
2530 LOCATE 2,22:PRINT "I HAVE BUST"
:GOSUB 4270
2540 LOCATE 2,23:PRINT "ANOTHER GAME
?(Y/N)"
2550 Y$=INKEY$:IF Y$<>"Y" AND Y$<>"N
" THEN 2550
2560 IF Y$="Y" THEN 2580
2570 IF Y$="N" THEN 2570
2580 LOCATE 6,8:PRINT "            "
2590 LOCATE 6,9:PRINT "            "
2600 LOCATE 6,14:PRINT "            "
2610 LOCATE 6,15:PRINT "            "
2620 LOCATE 20,8:PRINT "            
        "

2630 LOCATE 20,14:PRINT "           
         "

2640 LOCATE 27,10:PRINT "     "
2650 LOCATE 27,16:PRINT "     "
2660 LOCATE 2,22:PRINT "            
   "

2670 LOCATE 2,23:PRINT "            
         "

2680 GOTO 1000
```

In line 2080 the score (called HSCORE) is entered, and if it is more than 17 (the number usually chosen as the dividing line in pontoon) the computer prints its total, and (in line 2110) goes to 2350. There a comparison is made with YSCORE (Your score) and, with the three alternatives (Equal, More, or Less) moves to other lines that terminate that sub-game, and add to either of the main scores.

In lines 2090 to 2110 other alternatives are given (similar to our way of thinking when deciding to TWIST or STICK) that send it to other lines. At this stage it will pay you to study very carefully the way in which these lines are used. Do NOT forget line 2070, which is there to introduce a pause, so that the computer appears to be thinking things out. Without that pause line the computer would move so fast that the end would arrive before you had time to see what was happening.

Lines 2120 to 2320 are a series of repetitions of the moves, each for a different domino. When lines 2350 to 2370 are reached take particular note of how they are used: IF INT(HSCORE) = INT(YSCORE) THEN 2380. This ensures that the number of HSCORE is compared with the number of YSCORE; and for each comparison there is a different line to go to. A draw adds nothing to either game score, while 'I WIN' or 'YOU WIN' adds to the appropriate score. 'YOU BUST' in line 1410, and 'I HAVE BUST' in line 2530 similarly adds to the appropriate score.

There is a lot of information about how to program games into the Amstrad 464 in this chapter, and after it is safely on tape it would be a beneficial exercise to alter parts of it to suit your own ideas.

## Chapter Nine

# ORIGIN & DRAW USAGE WITH COLOURS

In the previous chapters there have been many examples of making U.D.Graphics; of using FOR/NEXT loops in various ways; some practice with INKEY$ for inputs; a little experience with variables; and the use of LOCATE/PRINT statements. ORIGIN and DRAW have been used, but not to any great extent, and it is time now to extend that part of your knowledge: not with a game this time, but with a program full of tips that will be well worth keeping for future reference.

ORIGIN and DRAW use the PIXEL coordinates: from left to right along the screen bottom (for columns) and numbered from 0 to 640 (the X numbers); and up the right-hand side (for rows) and numbered from 0 to 400 (the Y numbers). For instance: ORIGIN 320,208: is a spot near the screen centre. It must be remembered that PIXEL coordinates (their X,Y numbers) must always be written in that order: bottom (columns) and side (rows). Fix this most firmly in your mind.

There are many instances where a filled-in triangle (or one made with stripes) can be used to good effect when writing a program. But it is not easy to remember how a triangle that points in this (or that) direction can be achieved; for a considerable amount of mental juggling with the – and + signs is needed. Therefore a program incorporating different ways of doing this can be of great use. In this program the triangles are assembled to print on the screen with their own line numbers under each one, so that in future it will be easy to locate one at a glance. This program will be written in sections.

```
5 REM  *************
10 REM ORIGIN & DRAW
15 REM *************
20 INK 0,26:INK 1,0:INK 2,11:INK 3,6
25 BORDER 2:PAPER 0:PEN 1:CLS
```

```
30 FOR X=40 TO 0 STEP-4:ORIGIN 62-X,
382-X:DRAW +X,0,1:NEXT X
40 FOR X=0 TO 40 STEP 4:ORIGIN 76,38
2-X:DRAW X,0:NEXT X
50 FOR X=40 TO 0 STEP-4:ORIGIN 156,3
42+X:DRAW -X,0,2:NEXT X
60 FOR X=40 TO 0 STEP-4:ORIGIN 174,X
+342:DRAW X,0:NEXT X
70 FOR X=42 TO 0 STEP-6:ORIGIN X+214
,342:DRAW 0,X,1:NEXT X
80 FOR X=0 TO 42 STEP 6:ORIGIN 308-X
,342:DRAW 0,X:NEXT X
90 FOR X=40 TO 0 STEP-6:ORIGIN X+308
,382:DRAW 0,-X,2:NEXT X
100 FOR X=40 TO 0 STEP-6:ORIGIN 408-
X,382:DRAW 0,-X:NEXT X
110 FOR X=40 TO 0 STEP-4:ORIGIN 470-
X,382-X:DRAW X*2,0,3:NEXT X
120 FOR X=0 TO 40 STEP 4:ORIGIN 550-
X,342+X:DRAW X*2,0:NEXT X
125 LOCATE 3,5:PRINT "30 40 50 60 70
 80 90 100  110  120"
```

When this list is entered and RUN, it will produce on the screen a row of triangles made with lines. Whether the triangle is solid or made with lines depends entirely on the STEP value in the FOR statement. Numbers that run upward have STEP 4 (or 6); those that run downward have STEP–4 (or –6). If the STEP number is changed to 1 the triangles will be solid; they have been made with lines in this program to make it easier to understand how the triangle is constructed.

The second point about these STEP values has to do with this computer: some are STEP 4 and some are STEP 6. When the lines to be drawn are horizontal ones the STEP number is 4; and when vertical the STEP number is 6. This is because on the Amstrad (and this has been mentioned previously) the pixels are less wide than they are high. Therefore, to make them print on the screen as of equal distance apart, the difference in Step number is used. If this is not quite clear to you, then alter all the STEP 6 values to 4, and study the difference this makes.

A different point to make refers to the use of colour. It can be seen that the pairs of triangles are of different colours. This is easy to control with the Amstrad because the DRAW command has a third part to it with which to control its colour. This third part in the statement (after a comma) controls the colour. It must be one of those defined in the INK statement (line 20), and it will operate only (repeat ONLY) on the DRAW statements that follow. This can be seen by comparing the DRAW statements in all lines entered so far. Lines 30, 50, 70, 90, and 110 have this addition to the DRAW statement. This is a handy facility.

The manner in which the various + and – signs are used in ORIGIN and DRAW statements needs a more detailed explanation (and study). We will do this with an example using simple numbers: FOR X=0 TO 20 STEP 4: ORIGIN 100+X,100–X: DRAW X,0: NEXT X.

This draws a triangle sloping from a point at 100,100 down to the right; because the FOR NEXT loop controls it. It first draws a dot at 100,100: second is a line of 4 pixels at 100+4,100–4: the third is a line of 8 pixels at 100+8,100–8 (108,92): the fourth is a line of 12 pixels at 100+12,100–12 (112,88): the fifth is a line of 16 pixels at 100+16,100–16 (116,84); and so on until the FOR/NEXT loop reaches the final line (in this case) of 20 pixels at 100–20,100–20 (120, 80).

If you have followed this you will now be able to construct triangles of any shape. If you have not, then enter at line 5000 the formula given above, and RUN it after clearing the screen (CLS). It will appear at the bottom left-hand corner of the screen. Count the number of lines, including the dot: 0 TO 20 divided by 4 (the STEP value) gives 5, but including the original position (0) it is six: a dot, a line of 4 pixels, one of 8 pixels, and so on.

ENTER LIST 5000 (without clearing the screen) and the line will appear at the screen top without disturbing the triangle. Now go through the explanation again. It should be perfectly clear. Try altering the + and – signs (only one at a time) and study the result. With more experiments like this you will soon grasp the principles involved.

We will now enter the next row of triangles, for they introduce some other differences in managing DRAW statements.

```
130 FOR X=30 TO 0 STEP-6:ORIGIN 70-X
,280-X:DRAW 0,X*2,1:NEXT X
140 FOR X=0 TO 30 STEP 6:ORIGIN 104+
X,280-X:DRAW 0,X*2:NEXT X
150 FOR X=0 TO 56 STEP 4:ORIGIN 184-
X/3,306-X:DRAW X/3*2,0,2:NEXT X
160 FOR X=56 TO 0 STEP-4:ORIGIN 248-
X/3,250+X:DRAW X/3*2,0:NEXT X
170 FOR X=0 TO 52 STEP 6:ORIGIN 334-
X,272+X/3:DRAW 0,-X/3*2,3:NEXT X
180 FOR X=52 TO 0 STEP-6:ORIGIN 358+
X,272-X/3:DRAW 0,X/3*2:NEXT X
190 FOR X=0 TO 32 STEP 6:ORIGIN 452+
X,248+X:DRAW 0,X*1.25,1:NEXT X
200 FOR X=30 TO 0 STEP-6:ORIGIN 556-
X,248+X:DRAW 0,X*1.25:NEXT X
205 LOCATE 3,11:PRINT "130 140 150 1
60 170  180  190  200"
```

In line 130 and onward it will be seen that instead of a right-angled triangle we are now drawing equilateral triangles (like two right-angled triangles back to back). What happens is that the X multiplied by 2 does just that. In some cases a division sign is used also, and this makes the triangle much narrower. Take any of these lines and transfer them to 5000 to experiment with, and you will learn a lot, particularly with such complex lines as those at 170 and 180. They require studying quite a lot to see how they work. Lines 190 and 200, although they produce peculiarly shaped triangles, are more straightforward. It is time now to enter the final list in this section.

```
210 FOR X=54 TO 0 STEP-6:ORIGIN 82-X
,218-X:DRAW 0,X*0.5,3:NEXT X
220 FOR X=54 TO 0 STEP-6:ORIGIN 122+
X,218-X:DRAW 0,X*0.5:NEXT X
230 FOR X=0 TO 32 STEP 4:ORIGIN 188+
X,208-X:DRAW X*2,0,1:NEXT X
240 FOR X=32 TO 0 STEP-4:ORIGIN 388-
X,208-X:DRAW -X*2,0:NEXT X
250 FOR X=32 TO 0 STEP-4:ORIGIN 384+
X,174+X:DRAW X*2,0,2:NEXT X
260 FOR X=32 TO 0 STEP-4:ORIGIN 588-
X,174+X:DRAW -X*2,0:NEXT X
265 LOCATE 3,16:PRINT "210   220   2
30  240    250   260"
270 FOR X=0 TO 32 STEP 6:ORIGIN 52-X
,74:DRAW X,0+X*2,1:NEXT X
280 FOR X=0 TO 32 STEP 6:ORIGIN 102+
X,74:DRAW -X,0+X*2:NEXT X
290 FOR X=0 TO 32 STEP 6:ORIGIN 216-
X,74:DRAW -X,0+X*2,2:NEXT X
300 FOR X=0 TO 32 STEP 6:ORIGIN 258+
X,74:DRAW X,0+X*2:NEXT X
310 FOR X=0 TO 32 STEP 6:ORIGIN 360-
X,74:DRAW X*2,0+X*2,3:NEXT X
320 FOR X=0 TO 32 STEP 6: ORIGIN 430
+X,74:DRAW -X*2,0+X*2:NEXT X
330 FOR X=0 TO 20 STEP 4:ORIGIN 516-
X,72+X:DRAW X,0+X*2,1:NEXT X
340 FOR X=20 TO 0 STEP-4:ORIGIN 578+
X,72+X:DRAW -X,X*2:NEXT X
350 LOCATE 2,22:PRINT "270  280  290
  300  310  320  330 340"
360 LOCATE 2,24:PRINT "PRESS P FOR N
EXT SECTION"
370 Q$=INKEY$:IF Q$="" THEN 370
380 IF Q$="P" OR Q$="p" THEN CLS:GOT
O 1000
```

In these two rows of triangles are some with fairly involved instructions. Line 320, in particular, attracted my attention as worthy of use in a scene. It reminded me of the slanting sails used by boats on the River Nile. The DRAW command of that line is fairly complex, but easily handled. To make it bigger it would be necessary to alter only the FOR part of the command.

I shall use that, and some of the other triangles, in the next section we write, but before we move to that it would be really worthwhile if you took any lines that interested you, EDITed them by adding 5 in front of their line numbers to bring them into the 5000 section, and then carried out as many experiments as you could manage. The real way to master BASIC is to practice.

Lines 360 to 380 transfer the screen display to lines starting at 1000, and there we shall build a scene using triangles.

```
1000 REM ********************
1010 REM SCENE WITH TRIANGLES
1020 REM ********************
1030 INK 0,25:INK 1,20:INK 2,2:INK 3
,13
1040 BORDER 15:PAPER 0:PEN 3:CLS
1050 GOSUB 1500
1060 PEN 1:FOR X=1 TO 40:FOR Y=1 TO
3:LOCATE X,Y:PRINT B$:NEXT Y:NEXT X:
PEN 3
1070 PEN 2:FOR X=1 TO 40:FOR Y=15 TO
 22:LOCATE X,Y:PRINT A$:NEXT Y:NEXT
X:PEN 3
1080 FOR X=40 TO 0 STEP-2:ORIGIN 437
-X,362-X:DRAW X*2,0,3:NEXT X:REM PYR
AMID
1090 FOR X=60 TO 0 STEP-2:ORIGIN 524
-X,364-X:DRAW X*2,0:NEXT X:REM PYRAM
ID
1100 FOR X=20 TO 0 STEP-2:ORIGIN 566
```

```
-X,366-X:DRAW X*2,0:NEXT X:REM PIRAM
ID
1110 FOR X=0 TO 40:ORIGIN 165+X,120-
X:DRAW X*2,0,2:NEXT X:REM BOAT HULL
1120 FOR X=40 TO 0 STEP-2:ORIGIN 402
-X,120-X:DRAW -X*2,0:NEXT X:REM BOAT
 HULL
1130 FOR X=0 TO 20:ORIGIN 280-X*3,80
+X:DRAW X*6,0:NEXT X:REM BOAT DECK
1140 ORIGIN 214,106:DRAW 40,160,3:RE
M MAST
1150 FOR X=0 TO 110:ORIGIN 242+X,106
:DRAW -X*2,0+X*2,1:NEXT X:REM BOAT S
AIL
1160 LOCATE 2,11:PRINT G$" "C$D$"
"C$D$:LOCATE 2,12:PRINT H$" "E$F$"
 "E$F$
1170 LOCATE 18,11:PRINT C$D$"  "C$D$
" "G$:LOCATE 18,12:PRINT E$F$"  "E$F
$" "H$
1180 LOCATE 1,24:PRINT "PRESS T FOR
TRIANGLES"
1190 R$=INKEY$:IF R$="" THEN 1190
1200 IF R$="T" OR R$="t" THEN CLS:RU
N
1499 GOTO 1499
1500 SYMBOL AFTER 240
1510 SYMBOL 240,0,0,60,195,0,0,60,19
5:REM WAVES
1520 SYMBOL 241,85,0,170,0,85,0,170,
0:REM SKY
1530 SYMBOL 242,0,0,48,113,243,63,63
,31:REM CAMEL HEAD
1540 SYMBOL 243,0,96,240,248,254,255
,255,255:REM CAMEL RUMP
1550 SYMBOL 244,7,5,5,9,9,27,0,0:REM
 FRONT LEGS
```

```
1560 SYMBOL 245,14,10,10,18,18,54,0,
0:REM BACK LEGS
1570 SYMBOL 246,0,0,0,24,28,8,28,60:
REM MANS HEAD
1580 SYMBOL 247,60,60,124,124,124,40
,0,0:REM MANS BODY
1590 A$=CHR$(240):B$=CHR$(241):C$=CH
R$(242):D$=CHR$(243)
1600 E$=CHR$(244):F$=CHR$(245):G$=CH
R$(246):H$=CHR$(247)
1610 RETURN
```

In lines 1030 and 1040 we re-set the inks to be used, and also change the BORDER colour. At 1500 there is a section for U.D.G.s, because we need a sky colour, some water for the Nile, and some figures to place in the scene. Obviously there must be some pyramids, and these are drawn in lines 1080 to 1100, using the triangle from example on p.109 (line 110). The alterations made to this are to make them solid, to give them different sizes, and to locate them in different places; otherwise the formula used is the same.

For the boat and sail four different triangles are used: from lines 230, 240, 120, and 320. Note particularly the difference made to example 320 in line 1130: The X part of ORIGIN is multiplied by 3; and the X part of DRAW is multiplied by 6. This extends the triangle horizontally, and is quite apart from the differences caused by a different location.

The sail (line 1150) is made bigger than example 320 in the 'FOR' part of the statement. This is done by increasing the number '0 TO 32' to '0 TO 110', and the STEP section is left out to make the sail solid. In ORIGIN some different locations are used, of course, but the DRAW section is unchanged – except for the addition of the colour after the comma.

Take particular note that the mast is drawn (INK 3) before the sail, so that the light blue sail obliterates part of the mast as it is drawn. The light blue of the sail is the same as the colour used for the sky, but it looks different because the sky is made of dots, and allows the yellow paper to show through. This is one of the ways in which extra colours can be made.

### Colours

It has been mentioned previously that only four colours are available on the screen – including the one used for PAPER, the screen background. On page F3.7 of the User Instruction book is a program to demonstrate the colours. I found that one far too confusing, and so devised one of my own. Enter this now, using a new section that is on its own: use RUN 2000.

```
2000 REM ***********
2010 REM COLOUR TEST
2020 REM ***********
2030 FOR Z=26 TO 0 STEP-1
2040 FOR X=0 TO 26
2050 INK 0,26:INK 1,0:INK 2,X:INK 3,
Z
2060 BORDER X:PAPER 3:PEN 1:CLS
2070 LOCATE 10,8:PRINT "BORDER";X
2080 LOCATE 10,10:PRINT "PAPER";Z
2090 LOCATE 10,12:PRINT "PEN";X
2100 PEN 2: LOCATE 10,14:PRINT "THIS
 IS THE TEST":PEN 1
2110 FOR Y=1 TO 2000:NEXT Y
2120 NEXT X:NEXT Z
```

This, when entered and RUN, takes a considerable time to reach completion because of the pause caused by line 2110: FOR X=0 TO 2000. Change this to 200, and it runs through in just over five minutes; but to really study each colour, and how it could be used with others, replace that 2000: it is slower, and can easily be stopped by pressing ESC. Pressing the space bar re-starts it. That pause in line 2110 is completely within your control: reduced to 20 the sequence flashes through in about three minutes. If left at 2000 it takes over one minute to go through each PAPER colour (changing the BORDER and Test line 26 times).

Describing the actual colours produced by the Amstrad 464 is not too easy. The list on page F3.2 is no more than a general guide, for a little poetic licence must be allowed, but on the whole the colours are brilliant. However, two things have been attempted at the same time by the Amstrad programmers. They have catered for shades of greyness for the mono-colour monitors, and attempted to give a really good range of colours for the colour monitors.

Some of the colour monitors are evidently not adjusted too well, for the first one I received produced a few brownish-green colours near the middle of the range (at 9, 10, 12, and 13). However, a change of monitor soon made a big difference, and now I have only one criticism: 13, white, is really a mid-grey. By contrast most of the colours available are so bright and clear that one is constantly surprised. They give a wonderful display when used properly.

# Chapter Ten

## A THREE-DIMENSIONAL GAME

For this final chapter a game of Noughts and Crosses has been chosen. Not an ordinary game, but one with three sets of boards that are above one another, so that as well as having eight 3-in-line rows on a single board, there are also eight each on the other boards, giving 24; and in addition there can be more 3-in-line rows between the three boards: nine straight down, and sixteen diagonally down. This gives forty-nine possible 3-in-line results, and makes it increasingly difficult to plan moves ahead as the game progresses.

The game is designed for two players (not the computer) and our first task is to design the type of board display on which it will be played. After numerous attempts at drawing three boards to give an illusion of depth (3-D) I decided on square boards, one above the other, but tilted slightly, and with the nearest edge of each board thickened a little. The boards are viewed from slightly above. After this triple board was decided on, it was time to plan the program. Sections would be needed for X moves and O moves; one for the conversion of locations to X,Y numbers; a section for the screen display, and one for the graphics.

```
10 REM *********************
20 REM 3-D NOUGHTS & CROSSES
30 REM *********************
40 INK 0,25:INK 1,9:INK 2,14:INK 3,6
50 BORDER 18:PAPER 0:PEN 1:CLS
60 GOSUB 5000
70 GOSUB 4000
1000 REM *******
1010 REM X MOVES
1020 REM *******
1499 LOCATE 2,22:STOP
1500 REM *******
1510 REM 0 MOVES
1520 REM *******
```

```
1999 LOCATE 2,22:STOP
2000 REM ***********
2010 REM CONVERSIONS
2020 REM ***********
2999 LOCATE 2,22:STOP
4000 REM ******
4010 REM SCREEN
4020 REM ******
4520 RETURN
5000 REM ************
5010 REM U.D.GRAPHICS
5020 REM ************
5060 RETURN
```

Line 40 specifies the colours to be used: one for PAPER and three for PEN. Line 50 sets the BORDER at green, and the PAPER to pastel yellow. It is necessary now to form the graphic characters, X and O, and this is soon done in a very short section at 5000.

```
5030 SYMBOL AFTER 240
5040 SYMBOL 240,195,102,60,24,60,102
,195,0:REM X
5050 SYMBOL 241,60,126,195,195,195,1
26,60,0:REM O
```

With this entered we can continue with the designing of the triple board at 4000. The numbers for each position will be put in first (1 to 9, three times), and the board can then be drawn around them. The final structure is of three boards in layers, held in position by a solid board at the back, and by two stout supports at the front edges. This is contrived in lines 4040 to 4220, which we will enter now, together with a temporary STOP notice (4230 GOTO 4230) in order that we can study the result on the screen without the intrusion of a STOP notice.

```
4030 X=6:FOR Y=3 TO 17 STEP 7:LOCATE
 X,Y:PRINT "1 2 3":NEXT Y
4040 X=6:FOR Y=5 TO 19 STEP 7:LOCATE
 X,Y:PRINT "4 5 6":NEXT Y
4050 X=6:FOR Y=7 TO 21 STEP 7:LOCATE
 X,Y:PRINT "7 8 9":NEXT Y
4060 FOR X=54 TO 60 STEP 2:Y=54:ORIG
IN X,Y:DRAW 0,228,1:NEXT X
4070 FOR X=178 TO 184 STEP 2:Y=56:OR
IGIN X,Y:DRAW 0,224:NEXT X
4080 X=56:FOR Y=54 TO 58 STEP 2:ORIG
IN X,Y:DRAW 128,0:NEXT Y
4090 X=56:FOR Y=166 TO 170 STEP 2:OR
IGIN X,Y:DRAW 128,0:NEXT Y
4100 X=56:FOR Y=278 TO 282 STEP 2:OR
IGIN X,Y:DRAW 128,0:NEXT Y
4110 ORIGIN 56,56:DRAW 14,96:ORIGIN
56,168:DRAW 14,96:ORIGIN 56,280:DRAW
 14,96
4120 ORIGIN 182,56:DRAW -14,96:ORIGI
N 182,168:DRAW -14,96:ORIGIN 182,280
:DRAW -14,96
4130 ORIGIN 72,152:DRAW 94,0:ORIGIN
72,264:DRAW 94,0:ORIGIN 72,376:DRAW
94,0
4140 FOR X=72 TO 166 STEP 6:Y=154:OR
IGIN X,Y:DRAW 0,10,1:NEXT X
4150 FOR X=72 TO 166 STEP 6:Y=266:OR
IGIN X,Y:DRAW 0,10:NEXT X
4160 ORIGIN 70,152:DRAW 0,16,1:ORIGI
N 70,264:DRAW 0,16:ORIGIN 168,152:DR
AW 0,16:ORIGIN 168,264:DRAW 0,16
4170 ORIGIN 62,88:DRAW 114,0,2:ORIGI
N 68,118:DRAW 104,0
4180 ORIGIN 62,200:DRAW 114,0:ORIGIN
 68,230:DRAW 106,0
```

```
4190 ORIGIN 62,312:DRAW 114,0:ORIGIN
 68,342:DRAW 106,0
4200 ORIGIN 104,60:DRAW 0,92:ORIGIN
136,60:DRAW 0,92
4210 ORIGIN 104,172:DRAW 0,92:ORIGIN
 136,172:DRAW 0,92
4220 ORIGIN 104,284:DRAW 0,92:ORIGIN
 136,284:DRAW 0,92
```

Temporary line 4230 holds the computer in abeyance after it has printed the earlier entries; to get out of its influence press ESC twice. This will produce a BREAK/READY notice on the screen.

With the lines just entered we have drawn the structure of three boards, one above another. The two uprights of the front are made with four lines each – drawn with the aid of FOR/NEXT loops in lines 4060 and 4070. In lines 4080 to 4100 the board front edges (3 lines each) are similarly drawn. The rest of the lines to 4220 are straightforward ORIGIN/DRAW statements using the third factor after DRAW for colour where necessary.

The triple board as it now appears seems to be satisfactory, so we can proceed with placing a letter for each board: A, B, and C, and with entering the title, the instructions, and the score positions. The two score positions have different colours: red for X, and blue for O. This colour change will also be used for the X and O on the board while playing. Keeping a score is quite essential in this game, for it is difficult to keep a mental count when the 3-in-line numbers may be so high.

```
4230 LOCATE 3,5:PRINT "A":LOCATE 3,1
2:PRINT "B":LOCATE 3,19:PRINT "C"
4240 PEN 3:LOCATE 16,3:PRINT "3-DIME
NSIONAL":LOCATE 20,5:PRINT "NOUGHTS
& CROSSES":PEN 1
4250 PEN 2:FOR X=15 TO 37:LOCATE X,2
:PRINT "*":NEXT X:X=15:FOR Y=3 TO 5:
LOCATE X,Y:PRINT "*":NEXT Y:PEN 1
```

```
4260 PEN 2:FOR X=15 TO 37:LOCATE X,6
:PRINT "*":NEXT X:X=37:FOR Y=3 TO 5:
LOCATE X,Y:PRINT "*":NEXT Y:PEN 1
4270 LOCATE 16,8:PRINT "WINNER HAS M
OST LINES":LOCATE 22,10:PRINT "OF TH
REE"
4280 PEN 3:LOCATE 20,12:PRINT CHR$(2
40)" GOES FIRST":PEN 1
4290 LOCATE 16,14:PRINT "PRESS LETTE
R &":LOCATE 26,15:PRINT "THEN NUMBER
"
4300 PEN 3:LOCATE 19,18:PRINT "SCORE
 ";CHR$(240);"  :":PEN 1
4310 PEN 2:LOCATE 19,20:PRINT "SCORE
 ";CHR$(241);"  :":PEN 1
```

Again add a temporary line: at 4500 this time: 4500 GOTO 4500. This prevents the RETURN at 4520 functioning, and you can inspect the screen for mistakes; we all make them.

After erasing 4500 we must now move to section 2000, the conversions. In this game we have three sets of location: A, 1 to 9; B, 1 to 9; and C, 1 to 9. To persuade the computer to recognise these locations is relatively easy with the Amstrad 464. Section 2000 will do it, and can be used as a GOSUB position which will be called for by both X and O.

The entries at 2000 are tedious, but not too long, and can be written in by typing the first line of conversions (line 2030), and then EDITing it for the others. For each letter (A, B, C) the only alterations are to the line number, the Y$= number, and the X,Y location at the end of each line. Don't forget to change the A to B, and B to C for the other two sets of nine lines.

```
2030 IF X$="A" AND Y$="1" THEN LOCAT
E 6,3
2040 IF X$="A" AND Y$="2" THEN LOCAT
E 8,3
```

```
2050 IF X$="A" AND Y$="3" THEN LOCAT
E 10,3
2060 IF X$="A" AND Y$="4" THEN LOCAT
E 6,5
2070 IF X$="A" AND Y$="5" THEN LOCAT
E 8,5
2080 IF X$="A" AND Y$="6" THEN LOCAT
E 10,5
2090 IF X$="A" AND Y$="7" THEN LOCAT
E 6,7
2100 IF X$="A" AND Y$="8" THEN LOCAT
E 8,7
2110 IF X$="A" AND Y$="9" THEN LOCAT
E 10,7
2120 IF X$="B" AND Y$="1" THEN LOCAT
E 6,10
2130 IF X$="B" AND Y$="2" THEN LOCAT
E 8,10
2140 IF X$="B" AND Y$="3" THEN LOCAT
E 10,10
2150 IF X$="B" AND Y$="4" THEN LOCAT
E 6,12
2160 IF X$="B" AND Y$="5" THEN LOCAT
E 8,12
2170 IF X$="B" AND Y$="6" THEN LOCAT
E 10,12
2180 IF X$="B" AND Y$="7" THEN LOCAT
E 6,14
2190 IF X$="B" AND Y$="8" THEN LOCAT
E 8,14
2200 IF X$="B" AND Y$="9" THEN LOCAT
E 10,14
2210 IF X$="C" AND Y$="1" THEN LOCAT
E 6,17
2220 IF X$="C" AND Y$="2" THEN LOCAT
E 8,17
```

```
2230 IF X$="C" AND Y$="3" THEN LOCAT
E 10,17
2240 IF X$="C" AND Y$="4" THEN LOCAT
E 6,19
2250 IF X$="C" AND Y$="5" THEN LOCAT
E 8,19
2260 IF X$="C" AND Y$="6" THEN LOCAT
E 10,19
2270 IF X$="C" AND Y$="7" THEN LOCAT
E 6,21
2280 IF X$="C" AND Y$="8" THEN LOCAT
E 8,21
2290 IF X$="C" AND Y$="9" THEN LOCAT
E 10,21
2400 RETURN
```

This list cannot be tested until some moves are made, but provided you have entered it correctly there should be no trouble. Our next task is to start the moves at 1000.

```
1030 LOCATE 2,24:PRINT "MOVE X TO (L
ETTER)               "
1040 X$=INKEY$:IF X$="" THEN 1040
1050 IF X$="A" OR X$="B" OR X$="C" T
HEN 1060
1060 LOCATE 2,24:PRINT "MOVE X TO (N
UMBER)               "
1070 Y$=INKEY$:IF Y$="" THEN 1070
1080 GOSUB 2000
1090 PEN 3:PRINT CHR$(240):PEN 1:SOU
ND 1,50,20
```

When entered and RUN these lines will allow you to place one red X on any position, but then the computer will stop with an error message. Ignore that. If you now repeat RUN you can try a different location – with the same result, for those seven lines only deal with the location and placing of one X. We must continue this section to test it more completely.

```
1100 PEN 3:LOCATE 2,24:PRINT "IS THA
T A THREE-IN-LINE?(Y/N)     ":PEN 1
1110 Q$=INKEY$:IF Q$="" THEN 1110
1120 IF Q$<>"Y" AND Q$<>"N" THEN 110
0
1130 IF Q$="N" THEN 1500
1140 IF Q$="Y" THEN VX=VX+1:PEN 3:LO
CATE 29,18:PRINT VX:PEN 1
1150 FOR X=100 TO 50 STEP -5:SOUND 1
,X,10:NEXT X
1160 PEN 2:LOCATE 2,24:PRINT "IS THE
RE ANOTHER?(Y/N)            ":PEN 1
1170 R$=INKEY$:IF R$="" THEN 1170
1180 IF R$<>"Y" AND R$<>"N" THEN 116
0
1190 IF R$="N" THEN 1500
1200 IF R$="Y" THEN VX=VX+1:PEN 3:LO
CATE 29,18:PRINT VX:PEN 1
1210 FOR X=100 TO 50 STEP -5:SOUND 1
,X,10:NEXT X: GOTO 1160
```

This completes the moves by X, and now a test can be made, but only for one move. Press RUN and ENTER, and when the message appears at the screen bottom 'MOVE X TO (LETTER)' press either A, B, or C. Then the message will change slightly to 'MOVE X TO (NUMBER)'. Press any number (1 to 9) you choose, and a red X will appear there; if it doesn't then you should check all your entries.

With the X printed a red message will ask 'IS THAT A THREE-IN-LINE?(Y/N)'. Do NOT press the N, for there is nothing at 1500 for it to go to. Press Y, and a red 1 should appear in the score position, while a tune is played.

Then a blue message will appear 'IS THERE ANOTHER? (Y/N)'. This needs an explanation. In this game of Three-

Dimensional Noughts and Crosses it is possible for the entry of one X (or O) to complete more than one 3-in-line. For instance: if A1, A5, C5, and C9 all had an X, then one placed at B5 would complete two lines of 3-in-line: from A1 through B5 to C9; and from A5 through B5 to C5. It would also be possible for more than two 3-in-line rows to be completed, and so the final line in this listing goes back to repeat the blue question: 'IS THERE ANOTHER?'. Only when N is pressed does it go on to the moves for O.

To re-capitulate: after a move is made there will always be a red question: 'IS THAT A THREE-IN-LINE?'. If N is pressed the computer goes to the opponent's move. If Y is pressed it goes to the blue question: 'IS THERE ANOTHER?', and continues to loop back to that until the N is pressed.

Each time the Y is pressed, 1 is added to that score, and this can be seen in line 1140, where VX=VX+1, and is printed (with PEN 3) at the score position. This can happen in line 1120 also, so if you press Y you get 1 added to your score. It is easy to cheat, but that rarely gives any lasting satisfaction.

Study those lines from 1160 to 1210, for they give a good example of how to use a question input. Line 1170 keeps the computer waiting until a key is pressed. Line 1180 sends the computer back to the question asked until either Y or N is pressed. If N is pressed the computer goes to the O moves, and if Y is pressed it prints another score, plays a small tune, and goes back to the question again.

We are now ready to enter the listing for the O moves at 1500. These lines are a repeat of those from 1000 onward – but with differences here and there. I entered them by EDITing, but you may prefer to enter them directly.

```
1530 LOCATE 2,24:PRINT "MOVE O TO (L
ETTER)                "
1540 X$=INKEY$:IF X$="" THEN 1540
1550 IF X$="A" OR X$="B" OR X$="C" T
HEN 1560
1560 LOCATE 2,24:PRINT "MOVE O TO (N
UMBER)                "
1570 Y$=INKEY$:IF Y$="" THEN 1570
```

```
1580 GOSUB 2000
1590 PEN 2:PRINT CHR$(241):PEN 1:SO
ND 1,100,20
1600 PEN 3:LOCATE 2,24:PRINT "IS TH
T A THREE-IN-LINE?(Y/N)    ":PEN 1
1610 Q$=INKEY$:IF Q$="" THEN 1610
1620 IF Q$<>"Y" AND Q$<>"N" THEN 16
0
1630 IF Q$="N" THEN 1000
1640 IF Q$="Y" THEN VO=VO+1:PEN 2:L
CATE 29,20:PRINT VO:PEN 1
1650 FOR X=100 TO 50 STEP -5:SOUND
,X,10:NEXT X
1660 PEN 2:LOCATE 2,24:PRINT "IS TH
RE ANOTHER?(Y/N)                ":PEN 1
1670 R$=INKEY$:IF R$="" THEN 1670
1680 IF R$<>"Y" AND R$<>"N" THEN 16
0
1690 IF R$="N" THEN 1000
1700 IF R$="Y" THEN VO=VO+1:PEN 2:L
CATE 29,20:PRINT VO:PEN 1
1710 FOR X=100 TO 50 STEP -5:SOUND
,X,10:NEXT X: GOTO 1660
```

With this list entered the program is now complete, and can be played right through in order to test everything. The complexities of the various combinations of 3 -in-line will begin to appear. Even when playing against oneself (taking X and O in turn) the game is fascinating, and requires a great deal of thought before each move. For the first time Noughts and Crosses is a game of skill, though doubtless some bright fellow will work out a formula to always ensure a win if starting with X. However, there are so many alternatives to watch for that this will take a great deal of thinking out. The game will entertain you for a long time, and the more practice you put in the better able you will be to beat opponents.

This is the final instruction in this book. If you have entered each game in turn, beginning at Chapter One, and not leaving out Chapter Nine, you will by now have acquired some skill with the simpler ways to use BASIC to program the Amstrad. If you only remember to always plan a skeleton plot before starting each game you will be well on the way.

It must be emphasized once again that these games have been written in the simplest form of BASIC; with the knowledge gained from this book a large number of programs can be written. However, there is much more to be learned, for the Amstrad 464 has so many functions that the commands and features described in this book constitute less than a quarter of what can be done. Amstrad BASIC contains over 150 keywords, and we have used only a few of these. What we have done, though, is to build a sound foundation, and when this has been learnt, it will become much easier to add in other functions and commands.

The index contains a list of the functions and commands used in the previous chapters, so if you have forgotten one, a quick reference to the index should put you on the right track.

# INDEX

Page

BREAK READY . . . . . . . . . . . . . . . . . . . . . 44
CAT . . . . . . . . . . . . . . . . . . . . . . . . . . . . 37
CHESS . . . . . . . . . . . . . . . . . . . . . . . . . . 75
CHUCK-A-LUCK . . . . . . . . . . . . . . . . . . . 26
COLOURS . . . . . . . . . . . . . . . . . . . . . . . . 93–94, 110, 116–117
DOMINO PATIENCE . . . . . . . . . . . . . . . . 13
DOMINO PONTOON . . . . . . . . . . . . . . . . 93
DOTTED LINES . . . . . . . . . . . . . . . . . . . 40, 55
DRAUGHTS . . . . . . . . . . . . . . . . . . . . . . 38
DRAW . . . . . . . . . . . . . . . . . . . . . . . . . . 41
EDITING . . . . . . . . . . . . . . . . . . . . . . . . 7, 61, 113, 122
FOR/NEXT . . . . . . . . . . . . . . . . . . . . . . 9–11, 35, 43
GO Assn . . . . . . . . . . . . . . . . . . . . . . . . 71
GO . . . . . . . . . . . . . . . . . . . . . . . . . . . . 63–74
GOSUB/RETURN . . . . . . . . . . . . . . . . . . 13, 14, 47
GRAPHICS . . . . . . . . . . . . . . . . . . . . . . . 5
IF/THEN . . . . . . . . . . . . . . . . . . . . . . . . 47
INKEY$ . . . . . . . . . . . . . . . . . . . . . . . . . 44, 47, 67
INPUT . . . . . . . . . . . . . . . . . . . . . . . . . . 44, 126
LIST . . . . . . . . . . . . . . . . . . . . . . . . . . . 11, 19
LOCATE/PRINT . . . . . . . . . . . . . . . . . . . 6, 16
NESTING . . . . . . . . . . . . . . . . . . . . . . . . 10
ORIGIN/DRAW . . . . . . . . . . . . . . . . . . . 1, 108, 115
PIXELS . . . . . . . . . . . . . . . . . . . . . . . . . 20, 108
PLOT . . . . . . . . . . . . . . . . . . . . . . . . . . . 7, 40
PLOTTING BOARD . . . . . . . . . . . . . . . . 19, 28, 78, 95
PRINT . . . . . . . . . . . . . . . . . . . . . . . . . . 8, 65
PUNCTUATION . . . . . . . . . . . . . . . . . . . 2, 4
R.N.D. . . . . . . . . . . . . . . . . . . . . . . . . . . 22, 30
RANDOMIZE . . . . . . . . . . . . . . . . . . . . . 22, 30
REVERSI . . . . . . . . . . . . . . . . . . . . . . . . 51
SAVE . . . . . . . . . . . . . . . . . . . . . . . . . . . 7
SKELETON . . . . . . . . . . . . . . . . . . . . . . 13, 26, 38, 51, 63
SOUND . . . . . . . . . . . . . . . . . . . . . . . . . 9, 11, 31
STEP . . . . . . . . . . . . . . . . . . . . . . . . . . . 9, 43, 109
STOP . . . . . . . . . . . . . . . . . . . . . . . . . . . 44
SYMBOL . . . . . . . . . . . . . . . . . . . . . . . . 6, 27
THREE-D O & X . . . . . . . . . . . . . . . . . . 118
TRANSPARENT . . . . . . . . . . . . . . . . . . . 2, 7, 57, 67
TRIANGLES . . . . . . . . . . . . . . . . . . . . . 108–115
U.D.G.s . . . . . . . . . . . . . . . . . . . . . . . . . 4, 53, 76, 115
VARIABLES . . . . . . . . . . . . . . . . . . . . . . 33, 35

# *Notes*

# *Notes*

# OTHER BOOKS OF INTEREST

**BP112: A Z80 WORKSHOP MANUAL**
**E.A. Parr, B.Sc., C.Eng., M.I.E.E.**

Intended for people who wish to progress beyond the stage of BASIC programming to topics such as machine code and assembly language programming or need hardware details of the Z80 based computer.

Starting with a review of computer principles, the book describes typical machine code instructions followed by a detailed description of the Z80 instruction set. Assembly language programming is discussed with examples.

Also given are hardware details of the Z80 and the use of associated I/O devices such as UARTs, PIOs and CTCs.

This book is not purely a descriptive text, however, Z80 hex machine code and assembler instructions are given in tabular form, along with in/out connections for Z80 and associated devices. It will therefore also be a useful reference book for the more experienced user.

*192 pages* *1983*
*ISBN 0 85934 087 2* **£2.75**

**BP152: AN INTRODUCTION Z80 MACHINE CODE**
**R.A. & J.W. Penfold**

Home computers are equipped with built-in software that enables them to be easily programmed to do quite complex tasks. The price that is paid for this programming ease is a relatively slow running speed, far lower than the speed at which the computer is really capable of running. Machine code programming entails direct programming of the micro-processor without using a built-in high level computer language such as BASIC. This gives a vast increase in running speed, but is something that can only really be undertaken by someone who has a reasonable understanding of the micro-processor and some of the other hardware in the computer.

Machine code programming is not as difficult as one might think, and once a few simple concepts have been grasped it is actually quite straightforward (although admittedly never as quick and easy as using a high level language). This book takes the reader through the basics of micro-processors and machine code programming, and no previous knowledge of these is assumed. The micro-processor dealt with here is the Z80 which is not one of the most simple types, but is generally acknowledged as one of the most powerful 8 bit devices, and is by no means excessively difficult for beginners. The Z80, or in most cases now the faster version the Z80A, are used in many home computers, including several of the most popular machines such as the Sinclair ZX81 and ZX Spectrum, plus the Memotech MTX500 and MTX512 machines, and the Amstrad CPC464. A few simple demonstration programs that can be run on these computers are included in this book.

*144 pages* *1984*
*ISBN 0 85934 127 5* **£2.25**

**BP153: AN INTRODUCTION TO PROGRAMMING THE AMSTRAD CPC464**

**R.A. & J.W. Penfold**

The excellent hardware of the Amstrad CPC464 running with Locomotive BASIC go to make up an extremely potent and versatile machine and this book has been written to help the reader expand the potential of this powerful combination, with the minimum of difficulty

The authors adopt a step-by-step approach starting with the fundamentals and then moving on to more advanced topics, with many example programs being included to illustrate and clarify points

In a book of this size it is impossible to fully cover every aspect of a machine as complex as the Amstrad CPC464, but the authors have tried, as far as possible, to complement the information supplied by the manufacturer rather than just duplicate it.

The text is divided into the following chapters: 1. Variables & Arrays; 2. String Variables; 3. Decisions; 4. INPUT, PRINT & DATA 5. The Sound Generator; 6. Graphics 1 – Modes & Colours; 7. Graphic 2 – Animation; 8. Binary & Hex; 9. Interfacing; 10. Interrupts.

*144 pages* *1984*

*ISBN 0 85934 128 3* **£2.25**

Please note following is a list of other titles that are available in our range of Radio, Electronics and Computer Books.

These should be available from all good Booksellers, Radio Component Dealers and Mail Order Companies.

However, should you experience difficulty in obtaining any title in your area, then please write directly to the Publisher enclosing payment to cover the cost of the book plus adequate postage.

If you would like a complete catalogue of our entire range of Radio, Electronics and Computer Books then please send a Stamped Addressed Envelope to:–

BERNARD BABANI (publishing) LTD
THE GRAMPIANS
SHEPHERDS BUSH ROAD
LONDON W6 7NF
ENGLAND

| | | |
|---|---|---|
| 160 | Coil Design and Construction Manual | £1.95 |
| 202 | Handbook of Integrated Circuits (IC's) Equivalents & Substitutes | £1.95 |
| 205 | First Book of Hi-Fi Loudspeaker Enclosures | £0.95 |
| 208 | Practical Stereo and Quadrophony Handbook | £0.75 |
| 214 | Audio Enthusiasts Handbook | £0.85 |
| 219 | Solid State Novelty Projects | £0.85 |
| 220 | Build Your Own Solid State Hi-Fi and Audio Accessories | £0.85 |
| 221 | 28 Tested Transistor Projects | £1.25 |
| 222 | Solid State Short Wave Receivers for Beginners | £1.95 |
| 223 | 50 Projects Using IC CA3130 | £1.25 |
| 224 | 50 CMOS IC Projects | £1.35 |
| 225 | A Practical Introduction to Digital IC's | £1.75 |
| 226 | How to Build Advanced Short Wave Receivers | £1.95 |
| 227 | Beginners Guide to Building Electronic Projects | £1.95 |
| 228 | Essential Theory for the Electronics Hobbyist | £1.95 |
| RCC | Resistor Colour Code Disc | £0.20 |
| BP1 | First Book of Transistor Equivalents and Substitutes | £1.50 |
| BP2 | Handbook of Radio, TV, Ind & Transmitting Tube & Valve Equivalents | £0.60 |
| BP6 | Engineers and Machinists Reference Tables | £0.75 |
| BP7 | Radio and Electronic Colour Codes and Data Chart | £0.40 |
| BP14 | Second Book of Transistor Equivalents & Substitutes | £1.75 |
| BP24 | 52 Projects Using IC741 | £1.75 |
| BP27 | Chart of Radio, Electronic, Semi-conductor and Logic Symbols | £0.50 |
| BP28 | Resistor Selection Handbook | £0.60 |
| BP29 | Major Solid State Audio Hi-Fi Construction Projects | £0.85 |
| BP32 | How to Build Your Own Metal and Treasure Locators | £1.95 |
| BP33 | Electronic Calculator Users Handbook | £1.50 |
| BP34 | Practical Repair and Renovation of Colour TVs | £1.95 |
| BP36 | 50 Circuits Using Germanium, Silicon and Zener Diodes | £1.50 |
| BP37 | 50 Projects Using Relays, SCR's and TRIACS | £1.95 |
| BP39 | 50 (FET) Field Effect Transistor Projects | £1.75 |
| BP42 | 50 Simple L.E.D. Circuits | £1.50 |
| BP43 | How to Make Walkie-Talkies | £1.95 |
| BP44 | IC 555 Projects | £1.95 |
| BP45 | Projects in Opto-Electronics | £1.95 |
| BP47 | Mobile Discotheque Handbook | £1.35 |
| BP48 | Electronic Projects for Beginners | £1.95 |
| BP49 | Popular Electronic Projects | £1.95 |
| BP50 | IC LM3900 Projects | £1.35 |
| BP51 | Electronic Music and Creative Tape Recording | £1.95 |
| BP52 | Long Distance Television Reception (TV-DX) for the Enthusiast | £1.95 |
| BP53 | Practical Electronics Calculations and Formulae | £2.95 |
| BP54 | Your Electronic Calculator and Your Money | £1.35 |
| BP55 | Radio Stations Guide | £1.75 |
| BP56 | Electronic Security Devices | £1.95 |
| BP57 | How to Build Your Own Solid State Oscilloscope | £1.95 |
| BP58 | 50 Circuits Using 7400 Series IC's | £1.75 |
| BP59 | Second Book of CMOS IC Projects | £1.95 |
| BP60 | Practical Construction of Pre-amps, Tone Controls, Filters & Attenuators | £1.95 |
| BP61 | Beginners Guide To Digital Techniques | £0.95 |
| BP62 | Elements of Electronics – Book 1 | £2.95 |
| BP63 | Elements of Electronics – Book 2 | £2.25 |
| BP64 | Elements of Electronics – Book 3 | £2.25 |
| BP65 | Single IC Projects | £1.50 |
| BP66 | Beginners Guide to Microprocessors and Computing | £1.95 |
| BP67 | Counter, Driver and Numeral Display Projects | £1.75 |
| BP68 | Choosing and Using Your Hi-Fi | £1.65 |
| BP69 | Electronic Games | £1.75 |
| BP70 | Transistor Radio Fault-Finding Chart | £0.50 |
| BP71 | Electronic Household Projects | £1.75 |
| BP72 | A Microprocessor Primer | £1.75 |
| BP73 | Remote Control Projects | £1.95 |
| BP74 | Electronic Music Projects | £1.75 |
| BP75 | Electronic Test Equipment Construction | £1.75 |
| BP76 | Power Supply Projects | £1.95 |
| BP77 | Elements of Electronics – Book 4 | £2.95 |

| | | |
|---|---|---|
| BP78 | Practical Computer Experiments | £1.75 |
| BP79 | Radio Control for Beginners | £1.75 |
| BP80 | Popular Electronic Circuits – Book 1 | £1.95 |
| BP81 | Electronic Synthesiser Projects | £1.75 |
| BP82 | Electronic Projects Using Solar Cells | £1.95 |
| BP83 | VMOS Projects | £1.95 |
| BP84 | Digital IC Projects | £1.95 |
| BP85 | International Transistor Equivalents Guide | £2.95 |
| BP86 | An Introduction to BASIC Programming Techniques | £1.95 |
| BP87 | Simple L.E.D. Circuits – Book 2 | £1.35 |
| BP88 | How to Use Op-Amps | £2.25 |
| BP89 | Elements of Electronics – Book 5 | £2.95 |
| BP90 | Audio Projects | £1.95 |
| BP91 | An Introduction to Radio DXing | £1.95 |
| BP92 | Easy Electronics – Crystal Set Construction | £1.75 |
| BP93 | Electronic Timer Projects | £1.95 |
| BP94 | Electronic Projects for Cars and Boats | £1.95 |
| BP95 | Model Railway Projects | £1.95 |
| BP96 | C B Projects | £1.95 |
| BP97 | IC Projects for Beginners | £1.95 |
| BP98 | Popular Electronic Circuits – Book 2 | £2.25 |
| BP99 | Mini-Matrix Board Projects | £1.95 |
| BP100 | An Introduction to Video | £1.95 |
| BP101 | How to Identify Unmarked IC's | £0.65 |
| BP102 | The 6809 Companion | £1.95 |
| BP103 | Multi-Circuit Board Projects | £1.95 |
| BP104 | Electronic Science Projects | £2.25 |
| BP105 | Aerial Projects | £1.95 |
| BP106 | Modern Op-Amp Projects | £1.95 |
| BP107 | 30 Solderless Breadboard Projects – Book 1 | £2.25 |
| BP108 | International Diode Equivalents Guide | £2.25 |
| BP109 | The Art of Programming the 1K ZX81 | £1.95 |
| BP110 | How to Get Your Electronic Projects Working | £1.95 |
| BP111 | Elements of Electronics – Book 6 | £3.50 |
| BP112 | A Z-80 Workshop Manual | £2.75 |
| BP113 | 30 Solderless Breadboard Projects – Book 2 | £2.25 |
| BP114 | The Art of Programming the 16K ZX81 | £2.50 |
| BP115 | The Pre-Computer Book | £1.95 |
| BP116 | Electronic Toys Games & Puzzles | £2.25 |
| BP117 | Practical Electronic Building Blocks – Book 1 | £1.95 |
| BP118 | Practical Electronic Building Blocks – Book 2 | £1.95 |
| BP119 | The Art of Programming the ZX Spectrum | £2.50 |
| BP120 | Audio Amplifier Fault-Finding Chart | £0.65 |
| BP121 | How to Design and Make your Own P.C.B.s | £1.95 |
| BP122 | Audio Amplifier Construction | £2.25 |
| BP123 | A Practical Introduction to Microprocessors | £2.25 |
| BP124 | Easy Add-on Projects for Spectrum ZX81 & Ace | £2.75 |
| BP125 | 25 Simple Amateur Band Aerials | £1.95 |
| BP126 | BASIC & PASCAL in Parallel | £1.50 |
| BP127 | How to Design Electronic Projects | £2.25 |
| BP128 | 20 Programs for the ZX Spectrum & 16K ZX81 | £1.95 |
| BP129 | An Introduction to Programming the ORIC-1 | £1.95 |
| BP130 | Micro Interfacing Circuits – Book 1 | £2.25 |
| BP131 | Micro Interfacing Circuits – Book 2 | £2.25 |
| BP132 | 25 Simple Shortwave Broadcast Band Aerials | £1.95 |
| BP133 | An Introduction to Programming the Dragon 32 | £1.95 |
| BP134 | Easy Add-on Projects for Commodore 64 & Vic-20 | £2.50 |
| BP135 | The Secrets of the Commodore 64 | £2.50 |
| BP136 | 25 Simple Indoor and Window Aerials | £1.95 |
| BP137 | BASIC & FORTRAN in Parallel | £1.95 |
| BP138 | BASIC & FORTH in Parallel | £1.95 |
| BP139 | An Introduction to Programming the BBC Model B Micro | £2.50 |
| BP140 | Digital IC Equivalents and Pin Connections | £3.95 |
| BP141 | Linear IC Equivalents and Pin Connections | £3.95 |
| BP142 | An Introduction to Programming the Acorn Electron | £1.95 |
| BP143 | An Introduction to Programming the Atari 600XL and 800XL | £2.50 |
| BP144 | Further Practical Electronics Calculations and Formulae | £3.75 |
| BP145 | 25 Simple Tropical and M.W. Band Aerials | £1.95 |

**BERNARD BABANI BP159**

# How to Write Amstrad CPC 464 Games Programs

- This book is written as a step-by-step guide to help you write your own graphics games programs on the amazing Amstrad CPC 464 micro. The games start at the most simple level and progress to a 3-Dimensional game.

- By working through each program in the order given in the book it is hoped that within a fairly short time a better understanding of Locomotive BASIC and how to logically plan your own programs will result.

- Essential reading for all Amstrad CPC 464 users, particularly those who are new to programming.

GB £ NET +002.50

ISBN 0-85934-134-8

00250

£2.50

9 780859 341349

How to Write Amstrad CPC 464 Games Programs BP159 W. SIMISTER